Mobile

Mobile architecture, offline-first, push notifications, app store, mobile security

- Mobile Security
- Mobile Architecture Document
- Offline-First Strategy
- Push Notification Design
- App Store Submission Checklist
- Mobile Architecture

Mobile Security

Mobile Security

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}} **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Mobile Threat Model

Threat Actor	Goal	Attack Vector	Likelihood	Impact	Mitigation
Malicious app on device	Steal auth tokens	Shared storage access	Medium	High	Secure storage (Keychain/Keystore)
Network interceptor (MITM)	Intercept API calls	Rogue WiFi, proxy	Medium	High	Certificate pinning
Reverse engineer	Extract API keys, business logic	APK/IPA decompilation	Medium	Medium	Code obfuscation, no hardcoded secrets
Stolen/lost device	Access user data	Physical access	High	High	Biometric lock, data encryption, remote wipe
Rooted/jailbroken device	Bypass controls	OS privilege escalation	Low	High	Root/jailbreak detection
Malicious insider	Access user data	Source code access	Low	High	Secrets in vault, no PII in logs

Assets to protect:

1. Auth tokens (access + refresh)
2. User PII (name, email, payment info)
3. API keys and secrets
4. Cached sensitive data

---

2. Authentication

2.1 Biometric Authentication

Implementation: `{{expo-local-authentication | react-native-biometrics}}`

```
import * as LocalAuthentication from 'expo-local-authentication';

export async function authenticateWithBiometrics(): Promise<boolean> {
  const hasHardware = await LocalAuthentication.hasHardwareAsync();
  const isEnrolled = await LocalAuthentication.isEnrolledAsync();

  if (!hasHardware || !isEnrolled) {
    return promptPINFallback();
  }

  const result = await LocalAuthentication.authenticateAsync({
    promptMessage: 'Verify your identity',
    fallbackLabel: 'Use PIN',
    cancelLabel: 'Cancel',
    disableDeviceFallback: false,
  });

  return result.success;
}
```

Use cases for biometric auth:

- ☐ App unlock after backgrounding > `{{5 minutes}}`
- ☐ View sensitive data (payment info, full account number)
- ☐ Confirm high-value transactions (amount > `{{$100}}`)

☐ Change security settings

Fallback: PIN code → `{{expo-secure-store}}` stored hash (bcrypt, not reversible)

2.2 Session Management

Property	Value
Access token lifetime	`{{15 minutes}}`
Refresh token lifetime	`{{30 days}}`
Refresh token rotation	Yes — single use, new token issued on use
Session timeout (background)	App lock after `{{5 minutes}}` inactive
Max concurrent sessions	`{{3 devices}}`
Force logout triggers	Password change, suspicious activity, admin revocation

Token storage:

```
// CORRECT — encrypted secure storage
await SecureStore.setItemAsync('access_token', token, {
  keychainAccessible: SecureStore.WHEN_UNLOCKED_THIS_DEVICE_ONLY,
});

// WRONG — never use AsyncStorage or MMKV for tokens
// await AsyncStorage.setItem('access_token', token); // DO NOT USE
```

2.3 Token Storage (Secure Enclave)

Platform	Mechanism	Access Level
iOS	Keychain Services	`kSecAttrAccessibleWhenUnlockedThisDeviceOnly`
Android	Android Keystore	`BIOMETRIC_STRONG` or `DEVICE_CREDENTIAL`

Cannot be backed up: Both mechanisms are device-bound — tokens do not transfer to new devices.

---

3. Data Protection

3.1 Secure Storage Policy

Data Type	Allowed Storage	Forbidden Storage
Access token	Keychain/Keystore	AsyncStorage, MMKV, SQLite
Refresh token	Keychain/Keystore	AsyncStorage, MMKV
Encryption key	Keychain/Keystore	Any other storage
User PII	SQLite (encrypted)	AsyncStorage
Non-sensitive prefs	AsyncStorage / MMKV	—

3.2 Data Encryption at Rest

Database encryption: `{{SQLCipher for SQLite | Realm Encryption}}`

```
// SQLCipher initialization with key from Keystore
const encryptionKey = await SecureStore.getItemAsync('db_encryption_key');
const db = await SQLite.openDatabaseAsync('app.db', {
  key: encryptionKey,
});
```

Key generation: On first launch, generate 256-bit random key, store in Keychain/Keystore.

Encrypted fields (beyond DB encryption):

Field	Encryption	Key Source
Payment card (last 4 only stored)	N/A — tokenized	Stripe/Braintree
SSN / national ID	AES-256-GCM	Keychain/Keystore
Private messages	End-to-end (Signal protocol)	Derived keys

3.3 Sensitive Data in Memory

Rule	Implementation
Clear passwords after use	Overwrite string variable, trigger GC
No sensitive data in logs	Custom logger strips PII fields

Rule	Implementation
No sensitive data in crash reports	Sentry `beforeSend` hook scrubs payload
No sensitive data in analytics	Map user ID → hashed ID before sending

---

3.4 Screenshot Prevention

```
// iOS — prevent screenshots programmatically
// (Note: React Native does NOT expose this API — native module required)

// Android — prevent screenshots and screen recording
import { Platform } from 'react-native';
import { preventScreenCapture, allowScreenCapture } from 'expo-screen-capture';

useEffect(() => {
  if (Platform.OS === 'android') {
    preventScreenCapture();
  }
  return () => allowScreenCapture();
}, []);
```

Screens requiring screenshot prevention:

- ☐ Payment / card details screen
- ☐ Account balance screen
- ☐ Personal document viewer
- ☐ `{{OTHER_SENSITIVE_SCREEN}}`

3.5 Clipboard Protection

- ☐ Password fields: `secureTextEntry={true}` — disables clipboard by default on iOS
- ☐ Sensitive data programmatically copied: clear clipboard after `{{60 seconds}}`
- ☐ Custom clipboard hook: `useSensitiveCopy(value, clearAfterMs: 60000)`

---

4. Network Security

4.1 Certificate Pinning

Library: `{{react-native-ssl-pinning | OkHttp certificate pinner (Android native) | URLSession (iOS native)}}`

```
// Using react-native-ssl-pinning
const response = await fetch('https://api.domain.com/endpoint', {
  method: 'POST',
  pkPinning: true,
  sslPinning: {
    certs: ['api_cert_sha256_fingerprint'],
  },
  headers: { /* ... */ },
  body: JSON.stringify(payload),
});
```

Pinned endpoints:

- `api.{{domain.com}}` — primary API
- `auth.{{domain.com}}` — authentication

Certificate rotation process:

1. Pin BOTH current cert and backup cert simultaneously
2. Deploy app update with new cert added
3. After old cert expires, remove old cert from pins
4. Test rotation in staging first

Handling pin failures:

- Log as security event to backend
- Show user-facing error: "Secure connection failed"
- Do NOT fall back to unpinned connection

4.2 SSL/TLS Configuration

Requirement	Value
Minimum TLS version	TLS 1.2
Preferred TLS version	TLS 1.3
HTTP allowed	No (ATS enforced on iOS, enforce on Android)
Cipher suites	Forward-secrecy only (ECDHE, DHE)

4.3 Network Request Encryption

- ☐ All requests over HTTPS (enforced via ATS + Android network security config)
- ☐ Sensitive payloads encrypted at application layer (in addition to TLS): `{{Yes/No}}`
- ☐ Request signing implemented (HMAC): `{{Yes/No}}`

---

5. Application Security

5.1 Code Obfuscation

Platform	Tool	Status
Android	ProGuard / R8 (enabled by default in release)	`{{Done/TODO}}`
iOS	Bitcode + Swift compiler optimizations	`{{Done/TODO}}`
JavaScript	Hermes bytecode compilation	`{{Done/TODO}}`
JavaScript (extra)	`{{metro-react-native-babel-preset obfuscation}}`	`{{Done/TODO}}`

ProGuard rules: `android/app/proguard-rules.pro` **TODO:** Review ProGuard rules to ensure no over-stripping of reflection-dependent code.

5.2 Root / Jailbreak Detection

Library: `{{expo-device | react-native-jail-monkey | custom}}`

```
import JailMonkey from 'jail-monkey';

export function checkDeviceIntegrity(): SecurityResult {
  const isJailbroken = JailMonkey.isJailBroken();
  const isDebuggedBuild = JailMonkey.isDebuggedMode();
  const onExternalStorage = JailMonkey.isOnExternalStorage(); // Android

  return {
    compromised: isJailbroken || isDebuggedBuild,
```

```
    reason: isJailbroken ? 'jailbroken' : isDebuggedBuild ? 'debug' : 'clean',
  };
}
```

Response to compromised device:

- `{{Warn user and allow use with reduced functionality | Block access entirely | Log silently}}`
- Justification: `{{Explain decision}}`

5.3 Tamper Detection

- ☐ App bundle signature verification on launch
- ☐ Resource hash verification for critical assets
- ☐ Backend validates app version — block known compromised versions

---

5.4 Debug Detection

```
// Detect if app is running under debugger in production
if (__DEV__ === false && detectDebugger()) {
  logSecurityEvent('debugger_attached');
  terminateSession();
}
```

Production builds must have:

- ☐ `__DEV__` mode disabled
- ☐ Console.log stripped (Babel plugin: `transform-remove-console`)
- ☐ Flipper disabled
- ☐ Source maps NOT bundled with the app binary (upload to Sentry separately)

5.5 Reverse Engineering Prevention

Measure	Implementation
No hardcoded secrets	All secrets fetched from vault at runtime or injected via CI
No plain API keys in bundle	Obfuscated + fetched from secure config endpoint

Measure	Implementation
Sensitive logic on server	Business logic requiring security stays backend-side
Binary protection	ProGuard / R8 (Android), Bitcode stripping (iOS)

TODO: Run MobSF static analysis before each major release and review findings.

---

6. OWASP Mobile Top 10 Checklist

#	Risk	Status	Notes
M1	Improper Credential Usage	`{{Pass/Fail/N/A}}`	Tokens in Keychain, no hardcoded creds
M2	Inadequate Supply Chain Security	`{{Pass/Fail}}`	Dependency audit `npm audit`
M3	Insecure Authentication/Authorization	`{{Pass/Fail}}`	JWT validated server-side
M4	Insufficient Input/Output Validation	`{{Pass/Fail}}`	Zod validation on all inputs
M5	Insecure Communication	`{{Pass/Fail}}`	TLS + cert pinning
M6	Inadequate Privacy Controls	`{{Pass/Fail}}`	Data minimization, GDPR
M7	Insufficient Binary Protections	`{{Pass/Fail}}`	Obfuscation, no debug build in prod
M8	Security Misconfiguration	`{{Pass/Fail}}`	No dev configs in prod build
M9	Insecure Data Storage	`{{Pass/Fail}}`	Keychain/Keystore + encrypted DB
M10	Insufficient Cryptography	`{{Pass/Fail}}`	AES-256, SHA-256, no MD5/SHA-1

---

7. Security Testing Tools

Tool	Type	When	Output
MobSF (Mobile Security Framework)	Static analysis	Pre-release	Risk report
Frida	Dynamic analysis	Penetration test	Runtime behavior
Objection	Runtime analysis	Penetration test	Bypass checks

Tool	Type	When	Output
Burp Suite	Network proxy	Penetration test	API vulnerabilities
OWASP ZAP	Automated scan	CI/CD	Vulnerability report
`npm audit` / `yarn audit`	Dependency scan	Every PR	CVE report

Penetration test cadence: `{{Annual | Before each major release | Quarterly}}` **Last pentest date:** `{{DATE}}` **Next pentest date:** `{{DATE}}`

---

8. Compliance Requirements

Requirement	Applicable	Notes
GDPR (EU users)	`{{Yes/No}}`	User data deletion, export endpoints
CCPA (California users)	`{{Yes/No}}`	Do Not Sell option
COPPA (under 13)	`{{Yes/No}}`	Parental consent flow
PCI DSS (payment data)	`{{Yes/No}}`	Use PCI-compliant SDK (Stripe/Braintree) — never store card data
App Store privacy label	Yes	App Store Connect privacy questionnaire
Play Store data safety	Yes	Google Play Console data safety form
HIPAA (health data)	`{{Yes/No}}`	BAA with vendors, encryption at rest+transit

---

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			
Security Lead			
Legal			

Mobile Architecture Document

Mobile Architecture Document

> **Project:** Drop — Fintech Payment App **Version:** 0.1.0 **Date:** 2026-02-23
> **Author:** John (AI Director, ALAI) **Status:** In Review **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial draft — from ADR-011, MOBILE-APP.md, source code analysis

---

1. Framework & Rationale

Framework	Pros	Cons	Decision
React Native (Expo SDK 54)	JS/TS codebase reuse with Next.js web app, large ecosystem, OTA updates via EAS, managed workflow	Bridge overhead, larger binary (~25MB), Expo-compatible packages only	**Selected**
Flutter	High performance, consistent UI, strong typing	Dart language (no code sharing with React web app), no native OTA	Rejected
Swift (iOS native)	Best iOS performance, latest APIs	iOS only, separate codebase	Rejected
Kotlin (Android native)	Best Android performance, Material 3	Android only, separate codebase	Rejected

Selected: `Expo SDK 54 (managed workflow)` **Version:** Expo SDK 54, React Native latest, Expo Router v4 **ADR Reference:** `docs/architecture/adr/ADR-011-expo-mobile-framework.md`

Rationale: Drop's web app uses React 19 + Next.js. Expo enables sharing React components, hooks, types, and business logic between web and mobile — reducing duplication and bugs. BankID authentication requires opening a secure browser (`expo-web-browser`) and handling deep link callbacks (`drop://auth/callback`), which Expo provides natively. QR scanning is a core Drop feature requiring camera access — `expo-camera` provides built-in barcode scanning. OTA updates via EAS enable rapid hotfixes for financial apps without App Store review cycles. The AI-driven development team benefits from a single language (TypeScript) across all platforms.

Runtime environment:

- Min iOS: iOS 16
- Min Android: Android 10 (API 29)
- Target SDK: Android 34 / iOS 18

---

2. Project Structure

```
src/drop-mobile/
├── App.js                          # Expo entry (unused — Expo Router takes over)
├── app/
│   ├── _layout.js                  # Root Stack layout + font loading + splash screen
│   ├── index.js                    # Welcome screen (green bg, "drop." wordmark)
│   ├── login.js                    # Login screen (email + password)
│   ├── register.js                 # Registration (2-step: info, password)
│   ├── history.js                  # Transaction history (filter tabs, FlatList)
│   └── (tabs)/
│       ├── _layout.js              # Tab navigator (4 tabs)
│       ├── index.js                # Dashboard / Home (balance card, transactions)
│       ├── send.js                 # Send money (2-step: recipient + amount)
│       ├── scan.js                 # QR scanner (camera + payment flow)
│       └── profile.js              # Profile & settings (recipients, logout)
├── lib/
│   ├── api.js                      # API client (Bearer token auth)
│   └── theme.js                    # Theme constants (colors, fonts, spacing, radius)
└── assets/                         # Static assets (images, icons)
```

Key design decision: Expo Router is file-based routing — the directory structure maps directly to navigation routes.

---

3. Navigation Architecture

```
graph TD
    Root["Root Stack Navigator\n(app/_layout.js)"] --> Welcome["Welcome Screen\napp/index.js"]
    Root --> Login["Login Screen\napp/login.js"]
    Root --> Register["Register Screen\napp/register.js"]
    Root --> Tabs["Tab Navigator\app/(tabs)/_layout.js"]
    Root --> History["History Screen\app/history.js (modal)"]

    Tabs --> Home["Home Tab\app/(tabs)/index.js\n(Dashboard)"]
    Tabs --> Send["Send Tab\app/(tabs)/send.js\n(Send Money)"]
    Tabs --> Scan["Scan Tab\app/(tabs)/scan.js\n(QR Scanner)"]
    Tabs --> Profile["Profile Tab\app/(tabs)/profile.js"]
```

Navigation library: Expo Router v4 (file-based, built on React Navigation)

Deep link handling:

- URL scheme: `drop://`
- BankID callback: `drop://auth/callback?code=&state=`
- Library: `expo-linking`
- Config: `app.config.ts` → `scheme: "drop"`

Tab Navigator Configuration (4 tabs):

Tab	Label	Icon	Screen
index	Hjem	Unicode house emoji	Dashboard
send	Send	Unicode arrow emoji	Send Money
scan	QR	Unicode QR emoji	QR Scanner
profile	Profil	Unicode person emoji	Profile

Tab bar style: `backgroundColor: #FFFFFF`, `borderTopColor: #E5E7EB`, `height: 60` **Active tint:** `#0B6E35` (Forest Green), **Inactive:** `#9CA3AF`

Note: Mobile has 4 tabs. Web has 5 tabs (adds "Aktivitet/Kontoer"). Mobile shows bank balance on dashboard, not a separate screen.

Auth flow: On login success, token stored → navigate to `/(tabs)`. On logout → clear token → navigate to `/`.

---

4. Platform-Specific Considerations

Concern	iOS	Android	Solution
Back gesture	Swipe from edge	Back button + gesture	Expo Router handles both
Status bar	Overlaps content	Separate space	`SafeAreaView` from expo
Permissions model	Request at use time	Request at use time + Manifest	`expo-camera` permissions
Push notifications	APNs	FCM	`expo-notifications` (unified)
Keyboard behavior	Push up content	May or may not push	`KeyboardAvoidingView`
Font rendering	System fonts crisp	Sub-pixel differences	Custom font loading via `expo-google-fonts`
Haptics	UIFeedbackGenerator	Vibrator API	`expo-haptics` (future)
Secure storage	Keychain	Keystore	`expo-secure-store`
BankID browser	In-app secure browser	In-app secure browser	`expo-web-browser`
Camera / QR	AVFoundation	Camera2 API	`expo-camera`

---

5. Build Variants & Flavors

Variant	Bundle ID	API URL	Debug	Analytics	Push Env
Dev	`no.getdrop.app.dev`	`http://localhost:3000/api`	Yes	Off	Development
Staging	`no.getdrop.app.staging`	`https://drop-app-staging.vercel.app/api`	No	Off	Development
Production	`no.getdrop.app`	`https://drop-app.vercel.app/api`	No	Yes	Production

Environment variable handling: `expo-constants` via `app.config.ts`

API URL Detection (current implementation):

```
// lib/api.js
const API_URL = __DEV__
  ? "http://localhost:3000/api"
  : "https://drop-app.vercel.app/api";
```

6. Code Sharing Strategy

Shared with web app (`src/drop-app/`):

- TypeScript interfaces (`User`, `BankAccount`, `Transaction`)
- Business logic (age validation, XSS input sanitization)
- API endpoint paths
- Brand tokens (colors, spacing) — same values in `theme.js` and `globals.css`

Mobile-specific:

- React Native StyleSheet (vs Tailwind CSS on web)
- `lib/api.js` — Bearer token auth (vs httpOnly cookies on web)
- `lib/theme.js` — Native theme constants
- Expo-specific modules (`expo-camera`, `expo-web-browser`, etc.)
- 4-tab navigation (vs 5-tab on web)

Web-only:

- Next.js App Router
- shadcn/ui components
- Server Components
- httpOnly cookie auth
- Cards feature (feature-flagged on web, not present on mobile)
- Merchant dashboard (on web, not on mobile)

---

7. Screens Detail

Welcome (`app/index.js`)

- Full-screen green background (`#0B6E35`)
- "drop." wordmark in white Fraunces 700 font
- Headline: "Enklere betalinger. Lavere gebyrer."
- Two buttons: "Logg inn" (outline) → /login, "Opprett konto" (solid white) → /register

Login (`app/login.js`)

- White background
- Email + password fields with React Native `TextInput`

- "Logg inn" green button → `api.login(email, password)`
- On success: stores token in AsyncStorage, navigates to `/(tabs)`
- "Opprett konto" link → /register
- Pre-filled demo credentials in `__DEV__` mode

Register (`app/register.js`)

- 2-step flow with progress dots indicator (vs 4-step on web)
- **Step 1:** firstName, lastName, email, phone
- **Step 2:** password, confirmPassword
- `api.register(data)` → on success navigate to `/(tabs)`

Dashboard (`app/(tabs)/index.js`)

- Greeting: "Hei, {firstName}" with hand wave emoji
- Green balance card: "Total saldo" — formatted NOK amount
- Quick stats row: Sendt, Mottatt, Ventende (sent, received, pending counts)
- "Siste transaksjoner" section with `FlatList`
- Each transaction: direction icon (arrow up=sent, arrow down=received), name, date, amount with color coding
- "Se alle" link → /history
- Pull-to-refresh via `RefreshControl`

Send Money (`app/(tabs)/send.js`)

- 2-step flow (vs 4-step on web — simpler mobile UX)
- **Step 1:** Recipient name input + currency picker (5 currencies with flags)
  - Currencies: BAM (Bosnia), RSD (Serbia), PKR (Pakistan), TRY (Turkey), PLN (Poland)
- **Step 2:** Amount input (NOK) + conversion card showing exchange rate, 0.5% fee, "Mottaker får" calculated amount
- "Bekreft og send" → `api.sendRemittance(data)` → success screen

QR Scanner (`app/(tabs)/scan.js`)

- Camera placeholder (gray box with QR icon) — simulated in current version
- "Skann QR-kode" instruction text
- "Simuler skanning" button for demo
- Nearby merchants list (hardcoded: Ahmetov Kebab, Kafe Oslo, Narvesen)
- Payment flow: merchant info → amount input → confirm → success
- `api.payQR({ merchantId, amount })`

Transaction History (`app/history.js`)

- Filter tabs: Alle, Sendinger (remittance), QR (qr_payment)
- `FlatList` with transaction items (direction icon, name, date, amount)
- Pull-to-refresh
- Filter changes trigger re-fetch via `api.getTransactions({ type })`

Profile (`app/(tabs)/profile.js`)

- User info section: initials avatar, name, email
- "Mine mottakere" section with recipients list (fetched from `api.getRecipients()`)
- Each recipient: name, country, account number
- Settings menu: Sprak, Varsler, Personvern, Vilkar
- Logout button → clears token, navigates to `/index`

---

8. Native Module Integration

Module	Purpose	Drop Feature
`expo-camera`	Camera access + barcode scanning	QR payment scanning
`expo-web-browser`	Secure in-app browser	BankID OIDC authentication
`expo-notifications`	Push notification handling	Transaction alerts, payment receipts
`expo-linking`	Deep link handling (`drop://`)	BankID callback, notification deep links
`@react-native-async-storage/async-storage`	Persistent key-value store	Bearer token storage
`expo-secure-store`	Encrypted storage (Keychain/Keystore)	Sensitive data (future biometric)
`expo-local-authentication`	Biometric auth (Face ID/fingerprint)	App unlock — Phase 2
`expo-haptics`	Haptic feedback	Payment confirmation — Phase 2
`expo-google-fonts`	Font loading	Fraunces, DM Sans

New native module process:

1. Check if Expo SDK covers the need
2. Check community modules (well-maintained, TypeScript types)
3. Custom native module only as last resort
4. Any native module must have TypeScript wrapper

---

9. Performance Optimization Strategy

Strategy	Implementation	Status
JS thread optimization	Minimal computation — no heavy processing in current version	Done
Image caching	No images in current version — text-based UI	N/A
List performance	`FlatList` with `keyExtractor`	Done
Bundle size	Hermes engine enabled (Expo default)	Done
Font loading	`expo-google-fonts` — loads async, SplashScreen prevents rendering until ready	Done
Memory management	`useEffect` cleanup in API calls	Partial
Render optimization	No `React.memo` yet — simple screens don't need it	Planned

Performance targets:

- App cold start to interactive: < 2 seconds
- Screen transition: < 300ms (React Navigation default)
- List scroll: 60fps
- Bundle size: < 25MB (Expo managed workflow)

---

10. CI/CD for Mobile

```
flowchart LR
    PR["Pull Request"] --> UnitTests["Unit Tests\n(Jest)"]
    UnitTests --> Build["Build\n(EAS Build)"]
    Build --> Distribute["Distribute\n(TestFlight / Firebase App Distribution)"]
    Distribute --> QA["QA Approval"]
    QA --> Store["Store Submission\n(EAS Submit)"]
```

Stage	Tool	Trigger
Build	EAS Build	Push to `main` or `release/*`
OTA hotfix	EAS Update	Any JS-only change to production
Test distribution	TestFlight (iOS) / Firebase App Distribution (Android)	Every staging build
App Store submit	EAS Submit	Manual trigger (release manager)
Code signing	Expo managed credentials	Automated

EAS advantage: OTA updates allow hotfixes to JavaScript bundle without App Store review. Critical for financial apps.

---

11. Architecture Diagram

```
graph TB
    subgraph "Mobile App (Expo SDK 54)"
        Screens["Screens (7 screens)\nWelcome, Login, Register, Dashboard,\nSend, Scan,
Profile, History"]
        Nav["Expo Router v4\nStack + Tabs navigation"]
        State["Local State (useState)\nNo global state library"]
        APIClient["api.js — API Client\nBearer token auth"]
        AsyncStorage["AsyncStorage\nToken persistence"]
        SecureStore["expo-secure-store\nFuture: sensitive data"]
    end

    subgraph "Expo Native Modules"
        Camera["expo-camera\nQR scanning"]
        WebBrowser["expo-web-browser\nBankID auth"]
        Notifications["expo-notifications\nPush alerts"]
        Linking["expo-linking\ndrop:// deep links"]
        Biometrics["expo-local-authentication\nPhase 2 — app unlock"]
    end

    subgraph "Backend"
        HonoAPI["Hono v4 REST API\n/v1/* Bearer token auth"]
        BankID["BankID OIDC\nAuthentication"]
        OpenBanking["Open Banking (PSD2)\nAISP + PISP"]
    end

    Screens --> Nav
    Screens --> State
    State --> APIClient
    APIClient --> AsyncStorage
    APIClient --> HonoAPI
    Screens --> Camera
    Screens --> WebBrowser
    WebBrowser --> BankID
```

```
    BankID --> Linking
    HonoAPI --> OpenBanking
    Notifications --> Screens
```

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	
Mobile Lead			
Tech Lead			
Product Owner			

Offline-First Strategy

Offline-First Strategy

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Offline Capability Requirements

Feature	Offline Support	Priority	Notes
View cached content feed	Required	P1	Last 50 items
Create draft (saved locally)	Required	P1	Sync when online
Search (local cache only)	Partial	P2	Degraded — no remote results
User authentication	Not required	—	Login requires network
Push notifications	N/A	—	Requires network by nature
File uploads	Queue	P2	Upload when network returns
`{{FEATURE}}`	`{{Required/Partial/Not required}}`	`{{P1-P3}}`	`{{Notes}}`

Offline minimum viable experience:

>

> TODO: Define what the user should see/do when completely offline — blank screen, cached data, or read-only mode.

---

2. Local Storage Architecture

2.1 Database (Structured Data)

Selected database: `{{WatermelonDB | SQLite (expo-sqlite) | Realm | TinyBase}}`

Rationale:

> TODO: Explain why this DB was chosen (query capability, sync support, performance, bundle size).

Schema overview:

```
-- Example schema — expand per domain

CREATE TABLE users (
  id TEXT PRIMARY KEY,
  name TEXT NOT NULL,
  email TEXT UNIQUE NOT NULL,
  avatar_url TEXT,
  synced_at INTEGER,
  updated_at INTEGER NOT NULL
);

CREATE TABLE posts (
  id TEXT PRIMARY KEY,
  user_id TEXT REFERENCES users(id),
  title TEXT NOT NULL,
  body TEXT,
  status TEXT DEFAULT 'published',
  is_local_draft INTEGER DEFAULT 0,
  synced_at INTEGER,
  created_at INTEGER NOT NULL,
```

```
  updated_at INTEGER NOT NULL
);

CREATE TABLE sync_queue (
  id TEXT PRIMARY KEY,
  entity_type TEXT NOT NULL,
  entity_id TEXT NOT NULL,
  operation TEXT NOT NULL, -- 'create' | 'update' | 'delete'
  payload TEXT NOT NULL,   -- JSON
  retry_count INTEGER DEFAULT 0,
  created_at INTEGER NOT NULL
);
```

TODO: Define full schema for all entities.

2.2 File Storage

Type	Location	Max Size	Eviction
Downloaded images	`{{FileSystem.cacheDirectory}}/images/`	200 MB	LRU on cache full
Downloaded documents	`{{FileSystem.documentDirectory}}/docs/`	500 MB	Manual user delete
Queued upload files	`{{FileSystem.documentDirectory}}/uploads/`	1 GB	On successful upload
Temporary files	`{{FileSystem.cacheDirectory}}/tmp/`	50 MB	On app start

Library: `{{expo-file-system | react-native-fs}}`

2.3 Secure Storage

Data	Storage	Reason
Auth token	`{{expo-secure-store}}`	Encrypted, Keychain/Keystore
Refresh token	`{{expo-secure-store}}`	Encrypted
Encryption key (for local DB)	`{{expo-secure-store}}`	Never in plain storage
User preferences	`AsyncStorage`	Non-sensitive

Rule: Anything accessed without a password must NOT be in secure storage (breaks biometric auth flow).

3. Sync Protocol Design

```
sequenceDiagram
    participant App
    participant LocalDB
    participant SyncQueue
    participant API

    Note over App,API: Online sync cycle

    App->>LocalDB: Read local data (immediate)
    App->>SyncQueue: Queue local changes
    App->>API: Push: POST /sync/push {changes}
    API-->>App: Server-applied changes + conflicts
    App->>LocalDB: Apply server changes

    App->>API: Pull: GET /sync/pull?since={timestamp}
    API-->>App: Remote changes since last pull
    App->>LocalDB: Merge remote changes

    Note over App,API: Offline scenario

    App->>LocalDB: Read cached data
    App->>SyncQueue: Queue changes (persisted)
    Note over SyncQueue: Waits for connectivity

    Note over App,API: Reconnect

    SyncQueue->>API: Drain queue — push all pending
    API-->>App: Conflict resolution
    App->>LocalDB: Merge resolved state
```

3.1 Sync Strategy

Approach: `{{Bidirectional delta sync}}`

Property	Value

Protocol	REST + `{{GraphQL subscriptions / WebSocket for live}}`
Push endpoint	`POST /sync/push`
Pull endpoint	`GET /sync/pull?since={unix_ms}&entities={list}`
Sync identifier	Per-entity `updated_at` timestamp (server clock)
Pull delta	Only records changed since last sync cursor
Batch size	Max 100 records per push, 200 per pull

3.2 Conflict Resolution

Entity	Strategy	Rationale
User profile	Last Write Wins (server wins)	Single-user edit
Post drafts	Last Write Wins (client wins on local draft)	User owns draft
Settings	Merge (union)	Non-conflicting fields
Counters (likes, views)	Server-side CRDT	Concurrent increments
`{{Entity}}`	`{{LWW / CRDT / Manual / Server wins}}`	`{{Reason}}`

Conflict detection: Compare `updated_at` + server-assigned `version` counter.

Manual conflict flow (when required):

1. Server returns `409 Conflict` with both versions
2. App stores both versions in local DB
3. User presented with diff UI to choose version
4. Resolved version pushed back to server

3.3 Sync Frequency & Triggers

Trigger	Action	Conditions
App foreground	Pull sync	Network available
Mutation (create/update/delete)	Immediate push	Network available; else queue
`AppState` change: background → foreground	Full sync	> 5 min since last sync
Network restored	Drain sync queue	Any queued changes
Timer (background fetch)	Pull sync	`{{Every 15 min}}`

Trigger	Action	Conditions
Push notification received	Pull sync for affected entity	Notification type = 'data_update'

3.4 Partial Sync / Delta Sync

- Client stores `last_sync_cursor` per entity type (epoch milliseconds)
- Pull requests include `since` cursor — server returns only changed records
- Deleted records: server maintains soft-delete with `deleted_at` for 30 days
- Client applies deletions, then removes soft-deleted records from local DB

---

4. Sync Queue Management

Queue storage: SQLite `sync_queue` table (survives app restart)

Queue item schema:

```
interface SyncQueueItem {
  id: string;             // UUID
  entityType: string;     // 'post' | 'user' | etc.
  entityId: string;
  operation: 'create' | 'update' | 'delete';
  payload: object;        // Full entity data
  retryCount: number;
  maxRetries: number;     // 5
  createdAt: number;      // Unix ms
}
```

Drain strategy:

1. On network restore: drain queue in FIFO order
2. Batch up to 50 items per push request
3. On error: retry with exponential backoff (1s, 2s, 4s, 8s, 16s)
4. After `maxRetries`: move to dead letter queue, notify user

TODO: Define user notification UX for sync failures.

---

5. Network State Detection & Handling

Library: `{{@react-native-community/netinfo}}`

```
// Network state hook
export function useNetworkState() {
  const [isOnline, setIsOnline] = useState(true);
  const [connectionType, setConnectionType] = useState<string>('unknown');

  useEffect(() => {
    return NetInfo.addEventListener((state) => {
      setIsOnline(state.isConnected && state.isInternetReachable);
      setConnectionType(state.type);
    });
  }, []);

  return { isOnline, connectionType };
}
```

UI behavior per state:

State	UI Response
Offline	Banner: "You're offline — showing cached data"
Reconnected	Banner: "Back online — syncing..." (auto-dismiss 3s)
Slow connection	No extra UI (handle transparently)
Sync in progress	Subtle indicator (not blocking)

---

6. Data Flow: Online vs Offline

```
flowchart TD
    UserAction["User Action"] --> CheckNetwork{Network\nAvailable?}

    CheckNetwork -->|Yes| DirectAPI["Send to API\ndirectly"]
    DirectAPI -->|Success| UpdateLocal["Update local DB"]
    DirectAPI -->|Error| QueueAction["Queue action\n+ optimistic update"]

    CheckNetwork -->|No| QueueAction
    QueueAction --> UpdateLocal
    UpdateLocal --> UpdateUI["Update UI\n(optimistic)"]
```

```
    style UpdateUI fill:#d4edda
    style QueueAction fill:#fff3cd
```

---

7. Testing Strategy for Offline Scenarios

Test Type	Scope	Tool
Unit	Sync queue operations	Jest
Unit	Conflict resolution logic	Jest
Integration	DB read/write with mock network	Jest + in-memory DB
E2E	Full offline → reconnect flow	Detox / Maestro
Manual	Network conditions simulator	`Network Link Conditioner` (iOS), `tc` (Android emulator)

E2E offline test scenario:

1. Open app online — verify data loads
2. Enable airplane mode
3. Perform create/update/delete actions
4. Verify optimistic UI updates
5. Verify actions queued (inspect DB)
6. Disable airplane mode
7. Verify sync queue drains
8. Verify server data matches local state

---

8. Storage Limits & Data Eviction Policy

Storage Type	Soft Limit	Hard Limit	Eviction Strategy
SQLite DB	50 MB	200 MB	Evict records older than 30 days
Image cache	150 MB	300 MB	LRU eviction
Document cache	200 MB	500 MB	User prompt to clear
Total app storage	500 MB	1 GB	Warn user, offer cleanup

Low storage alert: When device storage < 500 MB free, reduce cache limits by 50%.

User-initiated cleanup: Settings → Storage → Clear Cache option.

9. Error Handling & User Feedback

Error	User Feedback	Recovery Action
Sync push failed	Toast: "Couldn't sync — will retry"	Auto-retry with backoff
Conflict detected	Modal: "Update conflict — please resolve"	Manual resolution flow
Queue overflow (>500 items)	Warning banner	Partial push, user notified
Local DB corruption	Alert: "Storage error — please reinstall"	Offer fresh install
Storage limit reached	Alert with cleanup CTA	User clears cache

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			
Backend Lead			
Product Owner			

Push Notification Design

Push Notification Design

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Architecture Overview

```
sequenceDiagram
    participant Backend
    participant NotifService as Notification Service\n(OneSignal / Firebase)
    participant APNs as APNs (iOS)
    participant FCM as FCM (Android)
    participant Device

    Backend->>NotifService: POST /notifications\n{userId, type, data}
    NotifService->>APNs: Push payload (iOS users)
    NotifService->>FCM: Push payload (Android users)
    APNs-->>Device: Deliver notification (iOS)
    FCM-->>Device: Deliver notification (Android)
    Device->>Backend: Delivery receipt (optional)
    Device->>Backend: Open/click event (analytics)
```

Push service: `{{OneSignal | Firebase Cloud Messaging (unified) | AWS SNS | Custom}}`

2. Provider Setup

2.1 APNs (iOS)

Property	Value
Auth method	`{{APNs Auth Key (.p8) — preferred over cert}}`
Key ID	`{{KEY_ID — from Apple Developer Portal}}`
Team ID	`{{TEAM_ID}}`
Bundle ID	`{{com.company.app}}`
Environment	Dev: Sandbox
Key location	`{{Vault reference — never commit}}`

Capabilities required in Xcode:

☑ Push Notifications

☑ Background Modes → Remote notifications

- `{{[x] Background Modes → Background fetch}}` (if using background sync)

2.2 FCM (Android)

Property	Value
Project ID	`{{firebase-project-id}}`
Server key	`{{Vault reference}}`
Sender ID	`{{SENDER_ID}}`
google-services.json	`android/app/google-services.json` (gitignored — CI injected)

Android Manifest additions:

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<!-- For Android 13+ — request at runtime -->
```

2.3 Unified Service Configuration

SDK: `{{expo-notifications | react-native-firebase | onesignal-react-native}}`

```
// src/services/notifications.ts — abstraction layer
export async function registerForPushNotifications(): Promise<string | null> {
  const { status: existingStatus } = await Notifications.getPermissionsAsync();
  let finalStatus = existingStatus;

  if (existingStatus !== 'granted') {
    const { status } = await Notifications.requestPermissionsAsync();
    finalStatus = status;
  }

  if (finalStatus !== 'granted') return null;

  const token = await Notifications.getExpoPushTokenAsync({
    projectId: Constants.expoConfig?.extra?.eas?.projectId,
  });

  // Register token with backend
  await api.post('/users/push-token', { token: token.data });
  return token.data;
}
```

---

3. Notification Types & Channels

3.1 Transactional Notifications

Type	Trigger	Priority	Sound	Badge
`order.confirmed`	Order placed	High	Default	+1
`payment.received`	Payment processed	High	Default	+1
`message.received`	New chat message	High	Custom	+1
`account.security`	Password changed, new login	Critical	Default	+1
`{{TYPE}}`	`{{TRIGGER}}`	`{{High/Normal/Low}}`	`{{Default/Custom/None}}`	`{{+1/None}}`

3.2 Marketing / Engagement Notifications

Type	Description	Frequency Cap	Opt-out Channel
`promo.offer`	Discount or limited-time offer	Max 2/week	Marketing channel
`feature.announce`	New feature announcement	Max 1/month	Marketing channel
`re_engagement`	Win-back inactive users	Max 1/week	Marketing channel

3.3 Android Notification Channels

```
// Create channels on app startup (Android 8+ / API 26+)
await Notifications.setNotificationChannelAsync('transactional', {
  name: 'Orders & Payments',
  importance: Notifications.AndroidImportance.HIGH,
  vibrationPattern: [0, 250, 250, 250],
  lightColor: '#FF231F7C',
  sound: 'default',
});

await Notifications.setNotificationChannelAsync('messages', {
  name: 'Messages',
  importance: Notifications.AndroidImportance.HIGH,
  sound: 'message.wav',
});

await Notifications.setNotificationChannelAsync('marketing', {
  name: 'Promotions & Updates',
  importance: Notifications.AndroidImportance.DEFAULT,
  sound: null,
});
```

---

4. Payload Format & Schema

4.1 Data Payload vs Notification Payload

Type	Shown by OS?	Customizable	Use when
Notification payload	Yes (auto)	Limited	Simple alerts
Data payload	No	Full control	App handles display

Type	Shown by OS?	Customizable	Use when
Combined	Yes + data	Full	Standard approach

Recommended approach: Send both — notification payload for guaranteed delivery, data payload for app logic.

4.2 Standard Payload Schema

```
{
  "notification": {
    "title": "{{Notification title}}",
    "body": "{{Notification body text}}",
    "image": "{{optional image URL}}"
  },
  "data": {
    "notificationType": "{{order.confirmed | message.received | etc.}}",
    "entityType": "{{order | message | user | etc.}}",
    "entityId": "{{UUID}}",
    "deepLinkPath": "{{/orders/123}}",
    "timestamp": "{{ISO_8601}}",
    "version": "1"
  },
  "apns": {
    "payload": {
      "aps": {
        "sound": "default",
        "badge": 1,
        "content-available": 1
      }
    }
  },
  "android": {
    "channelId": "{{transactional | messages | marketing}}",
    "priority": "high"
  }
}
```

4.3 Localization

- All notification text must be localized
- Approach: `{{Backend sends localized text based on user's locale preference | App localizes using data payload keys}}`
- Fallback locale: `en`

---

5. Deep Linking Strategy

5.1 URL Scheme Configuration

URL scheme: `{{appname://}}` **Universal links (iOS):** `{{https://app.domain.com}}` **App links (Android):** `{{https://app.domain.com}}`

5.2 Deep Link Routing

Notification Type	Deep Link	Screen
`order.confirmed`	`appname://orders/{{orderId}}`	Order Detail
`message.received`	`appname://chats/{{chatId}}`	Chat Screen
`payment.received`	`appname://wallet`	Wallet Screen
`promo.offer`	`appname://offers/{{offerId}}`	Offer Detail

5.3 Navigation on Notification Tap

```
// App foreground — notification tap handler
Notifications.addNotificationResponseReceivedListener((response) => {
  const { notificationType, deepLinkPath } = response.notification.request.content.data;

  if (deepLinkPath) {
    // Use router.push for Expo Router, or navigation.navigate for React Navigation
    router.push(deepLinkPath);
  }
});

// App killed — check initial notification
useEffect(() => {
  Notifications.getLastNotificationResponseAsync().then((response) => {
    if (response?.notification.request.content.data?.deepLinkPath) {
      router.replace(response.notification.request.content.data.deepLinkPath);
```

```
    }
  });
}, []);
```

---

6. Opt-In / Opt-Out Flow

Permission request timing: `{{After onboarding, on first meaningful action — NOT on app launch}}`

Permission flow:

```
App Launch
    ↓
Onboarding Complete
    ↓
First Relevant Action (e.g., order placed)
    ↓
Pre-prompt Modal (custom UI — explain value)
    "Get notified when your order is ready"
    [Allow]  [Not now]
    ↓ (Allow)
OS Permission Dialog
    ↓
Register token with backend
```

Soft prompt: Always show custom pre-prompt before OS dialog. Users who dismiss OS dialog permanently cannot be re-asked — use pre-prompt to qualify intent.

---

7. Notification Preferences Per User

```
interface NotificationPreferences {
  pushEnabled: boolean;       // Master toggle
  channels: {
    transactional: boolean;   // Orders, payments — default: true
    messages: boolean;        // Chat messages — default: true
    marketing: boolean;       // Promotions — default: false
    security: boolean;        // Security alerts — always true, non-toggleable
```

```
  };
  quietHours: {
    enabled: boolean;
    start: string;            // "22:00" HH:mm
    end: string;              // "08:00" HH:mm
    timezone: string;         // "Europe/Oslo"
  };
}
```

API endpoint: `PUT /users/notification-preferences`

Sync: Preferences stored on server — synced on login and app foreground.

---

8. Rate Limiting & Throttling

Category	Limit	Window	Behavior at limit
Transactional	Unlimited	—	Always delivered
Messages	100	1 hour	Batch: "X new messages"
Marketing	2	7 days	Drop excess, no queue
Security	Unlimited	—	Always delivered

Backend enforcement: Rate limit checked before sending to push service. Excess notifications are logged but not sent.

---

9. Analytics & Tracking

Event	Tracked How	Data Points
Sent	Backend log	notificationType, userId, timestamp
Delivered	Push service receipt	notificationType, userId, deliveredAt
Opened	App handler	notificationType, userId, openedAt, timeToOpen
Dismissed	`{{Platform support}}`	`{{Android only – iOS limited}}`
Action tap	App handler	notificationType, actionId, userId

Funnel metrics:

- Delivery rate = delivered / sent
- Open rate = opened / delivered
- CTR (click-through) = action taps / opened

Privacy: User IDs are hashed in analytics. No PII in event data.

---

10. Testing Strategy

Test Type	Method	Environment
Payload validation	Unit test push service	Dev
Delivery smoke test	Send test notification via dashboard	Staging
Deep link routing	E2E test: tap notification → verify screen	Staging
Opt-in flow	E2E test: full permission flow	Staging / device
Rate limiting	Integration test: exceed limit → verify drop	Staging

Test push tool: `{{OneSignal Dashboard | Expo push notification tool | Firebase console}}`

TODO: Create Maestro test flows for notification tap → deep link scenarios.

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			
Backend Lead			
Product Owner			

App Store Submission Checklist

App Store Submission Checklist

> **Project:** {{PROJECT_NAME}} **Version:** {{APP_VERSION}} **Build:** {{BUILD_NUMBER}} **Date:** {{DATE}} **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

Pre-Submission Requirements

General Readiness

- ☐ All P1 and P2 bugs resolved — issue tracker link: `{{URL}}`
- ☐ QA sign-off obtained — sign-off document: `{{URL}}`
- ☐ Legal sign-off obtained (privacy policy, terms) — `{{URL}}`
- ☐ Release notes written and reviewed
- ☐ Version number follows `{{SemVer: MAJOR.MINOR.PATCH}}`
- ☐ Build number incremented (monotonically — never reused)
- ☐ All environment variables set for production
- ☐ Crash-free rate > 99.5% in staging (Sentry)
- ☐ Analytics verified — events firing correctly in staging

Apple App Store

App Store Connect Setup

- [ ] App record created in App Store Connect
- [ ] App ID registered in Apple Developer Portal
- [ ] Capabilities match Xcode project: `{{list capabilities used}}`
- [ ] Provisioning profiles up to date (distribution profile)
- [ ] Code signing certificate valid (not expiring within 30 days)
- [ ] App Store Connect API key configured for CI/CD submission

---

App Metadata

- [ ] **App name:** `{{App name}}` (max 30 chars)
- [ ] **Subtitle:** `{{Subtitle}}` (max 30 chars) — highlights key feature
- [ ] **Description:** `{{Description}}` (max 4000 chars) — engaging, keyword-rich
- [ ] **Keywords:** `{{keyword1, keyword2, ...}}` (max 100 chars total, comma-separated)
- [ ] **Promotional text:** `{{Promo text}}` (max 170 chars) — can update without new build
- [ ] **Support URL:** `{{https://support.domain.com}}`
- [ ] **Marketing URL:** `{{https://domain.com}}`
- [ ] **Privacy policy URL:** `{{https://domain.com/privacy}}`
- [ ] **Age rating** completed (4+ / 12+ / 17+)
- [ ] **Category:** Primary: `{{Category}}` | Secondary: `{{Category}}`
- [ ] **Copyright:** `{{Year}} {{Company Name}}`

Screenshots

Device	Dimensions	Required	Status
iPhone 6.7" (15 Pro Max)	1320×2868	Required	`{{Done/TODO}}`
iPhone 6.5" (11 Pro Max / 12 Pro Max)	1242×2688	Required	`{{Done/TODO}}`
iPhone 5.5" (8 Plus)	1242×2208	Required	`{{Done/TODO}}`
iPad Pro 12.9" (6th gen)	2048×2732	Required if iPad supported	`{{Done/TODO}}`

Device	Dimensions	Required	Status
iPad Pro 12.9" (2nd gen)	2048×2732	Required if iPad supported	`{{Done/TODO}}`

Screenshot rules:

- ☐ Max 10 screenshots per device
- ☐ First screenshot = most compelling (primary impression)
- ☐ No device frames required (add if chosen)
- ☐ No "Download on the App Store" badge in screenshots
- ☐ Text overlays readable at thumbnail size
- ☐ No third-party IP without permission

App Preview video (optional):

- ☐ Max 30 seconds, format MP4 or MOV
- ☐ Actual app footage — no simulated/demo content

App Privacy Details

- ☐ **Data types collected** declared — mapped to usage purpose:

Data Type	Collected?	Linked to User?	Used for Tracking?
Name	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Email	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Phone	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Location (precise)	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Location (coarse)	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Usage data	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Crash data	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`
Identifiers (device ID)	`{{Yes/No}}`	`{{Yes/No}}`	`{{Yes/No}}`

- ☐ App Tracking Transparency (ATT) framework implemented if using IDFA
- ☐ NSUserTrackingUsageDescription string provided in Info.plist

---

Review Guidelines Compliance

- [ ] No private API usage (review using `{{otool | MachOView}}`)
- [ ] No undocumented device capabilities
- [ ] In-app purchase implemented for digital goods (not bypassing IAP)
- [ ] External payment links removed or comply with court order rules (US only)
- [ ] Login options: if Sign in with Apple is available elsewhere, it MUST be offered
- [ ] User account deletion implemented (required since June 2022)
- [ ] App functions as described — demo account provided for review if needed

Demo account for App Review:

- Username: `{{review@domain.com}}`
- Password: `{{vault reference}}`
- Notes to reviewer: `{{special setup instructions}}`

TestFlight Beta Testing

- [ ] Internal testing completed (team members — up to 100)
- [ ] External beta testing completed — testers: `{{N}}`, duration: `{{N days}}`
- [ ] Crash rate < 1% in TestFlight
- [ ] Beta feedback addressed
- [ ] What's New in This Version: `{{Beta release notes}}`

App Transport Security

- [ ] All network connections use HTTPS
- [ ] No `NSAllowsArbitraryLoads: true` (or justified with `NSExceptionDomains`)
- [ ] Certificate pinning active for critical endpoints
- [ ] ATS exceptions documented: `{{list any exceptions and justification}}`

Common iOS Rejection Reasons — Prevention

Risk	Prevention
Crashes on launch	Test on physical device, clean install
Misleading screenshots	Screenshots match actual app UI

Risk	Prevention
Login required without guest mode	Provide review demo account
Missing privacy strings	All `NS*UsageDescription` keys populated
IAP bypass	All digital content purchases go through IAP
Placeholder content	Remove all Lorem Ipsum, test data
Performance issues on older devices	Test on min supported device

---

Google Play Store

Google Play Console Setup

- [ ] App created in Google Play Console
- [ ] Signing key configured (Play App Signing — recommended)
- [ ] Service account configured for CI/CD API access
- [ ] Developer account in good standing

Store Listing

- [ ] **App name:** `{{App name}}` (max 50 chars)
- [ ] **Short description:** `{{Short desc}}` (max 80 chars)
- [ ] **Full description:** `{{Full description}}` (max 4000 chars)
- [ ] **App icon:** 512×512 PNG, no alpha, no rounded corners (Play adds them)
- [ ] **Feature graphic:** 1024×500 JPG/PNG — shown at top of listing
- [ ] **Screenshots:** min 2, max 8 per device type

Device	Min Dimensions	Status
Phone	320×568 (min), 3840×3840 (max)	`{{Done/TODO}}`
7" tablet	Same constraints	`{{Done/TODO}}`
10" tablet	Same constraints	`{{Done/TODO}}`

Content Rating Questionnaire

- [ ] IARC questionnaire completed in Play Console
- [ ] Rating certificate generated and applied
- [ ] Rating matches app content (honest answers — inaccurate rating = suspension)

Data Safety Form

- [ ] Data types collected declared
- [ ] Data sharing disclosures complete
- [ ] Security practices answered:
  - [ ] Data in transit encrypted: `{{Yes/No}}`
  - [ ] Data at rest encrypted: `{{Yes/No}}`
  - [ ] Users can request deletion: `{{Yes/No}}`

Target Audience & Content

- [ ] Target age group declared (under 13? — COPPA compliance required)
- [ ] Ads configuration (if using ads) — appropriate ad formats for age group
- [ ] Sensitive app permissions justified in declaration

Testing Tracks

Track	Audience	Status
Internal testing	Up to 100 internal testers	`{{Done/TODO}}`
Closed testing (alpha)	Limited testers, feedback	`{{Done/TODO}}`
Open testing (beta)	Public opt-in	`{{Done/TODO}}`
Production	100% rollout / staged	`{{Done/TODO}}`

Staged rollout: Start at `{{10%}}` → increase after `{{48 hours}}` → `{{100%}}`

Common Android Rejection Reasons — Prevention

Risk	Prevention
Permission over-declaration	Request only necessary permissions
Misleading app behavior	App does exactly what listing says
Policy violations (ads)	No interstitials on back press, no deceptive ads
Malware detection	Scan APK/AAB with VirusTotal before upload
Crashes	Test on multiple API levels, both ARM architectures
Data safety inaccurate	Audit all SDKs for data collection

Cross-Platform Checklist

Version Naming

Field	iOS	Android	Value
Version string	`CFBundleShortVersionString`	`versionName`	`{{X.Y.Z}}`
Build number	`CFBundleVersion`	`versionCode`	`{{N}}` (monotonic)

Version naming convention: `MAJOR.MINOR.PATCH`

- MAJOR: Breaking change / major redesign
- MINOR: New feature
- PATCH: Bug fix / performance

Release Notes Format

```
What's new in v{{X.Y.Z}}:

• {{New feature 1}}
• {{Bug fix 1}}
• {{Improvement 1}}

Questions or feedback? Contact us at support@{{domain.com}}
```

Rules:

- Max 500 characters (App Store) / 500 characters (Play Store)
- Translate for each supported locale

- No marketing language — factual changes only
- Reference to known issues if applicable

Marketing Assets

- ☐ App icon final (no placeholder)
- ☐ Feature graphic final (Google Play)
- ☐ Press kit updated: `{{URL}}`
- ☐ App preview video (if applicable)
- ☐ Social media announcement content prepared

Legal Requirements

- ☐ **Privacy Policy** URL: `{{URL}}` — covers all data collected
- ☐ **Terms of Service** URL: `{{URL}}`
- ☐ GDPR: Right to deletion implemented
- ☐ CCPA: Do Not Sell link (if US users)
- ☐ COPPA: Kids category compliance (if < 13)
- ☐ In-app purchase terms linked

Final Submission Sign-Off

Item	Status	Sign-Off
All checklist items complete	`{{Yes/No}}`	
QA approval received	`{{Yes/No}}`	
Legal approval received	`{{Yes/No}}`	
Marketing assets ready	`{{Yes/No}}`	
Support team briefed	`{{Yes/No}}`	

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			
QA Lead			
Product Manager			
Legal			

Mobile Architecture

Mobile Architecture Document

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Framework & Rationale

Framework	Pros	Cons	Decision
React Native (Expo)	JS codebase reuse, large ecosystem, OTA updates	Bridge overhead, some native gaps	`{{Selected / Rejected}}`
Flutter	High performance, consistent UI, strong typing	Dart language, binary size	`{{Selected / Rejected}}`
Swift (iOS native)	Best iOS performance, latest APIs	iOS only, Swift/ObjC required	`{{Selected / Rejected}}`
Kotlin (Android native)	Best Android performance, Material 3	Android only	`{{Selected / Rejected}}`

Selected: `{{FRAMEWORK}}` **Version:** `{{VERSION}}` **Rationale:**

> TODO: 3-5 sentences explaining the final decision including team skills and project requirements.

Runtime environment:

- Min iOS: `{{iOS 16}}`
- Min Android: `{{Android 10 (API 29)}}`
- Target SDK: `{{Android 34 | iOS 17}}`

---

2. Project Structure

```
{{PROJECT_NAME}}/
├── src/
│   ├── app/                # Expo Router routes (file-based) or React Navigation config
│   ├── screens/            # Full-screen components
│   │   ├── auth/
│   │   ├── home/
│   │   └── settings/
│   ├── components/
│   │   ├── ui/             # Design system primitives
│   │   └── features/       # Feature-specific components
│   ├── navigation/         # Navigator definitions, types
│   ├── hooks/              # Shared custom hooks
│   ├── services/           # API client, native integrations
│   ├── stores/             # State management slices
│   ├── utils/              # Pure helpers
│   ├── constants/          # App-wide constants
│   └── types/              # TypeScript interfaces
├── ios/                    # iOS native code (Xcode project)
├── android/                # Android native code (Gradle project)
├── assets/                 # Images, fonts, icons
├── app.config.ts           # Expo config or equivalent
└── package.json
```

TODO: Update to match actual project structure.

---

3. Navigation Architecture

```
graph TD
    Root["Root Navigator"] --> Auth["Auth Stack\n(Unauthenticated)"]
    Root --> App["App Navigator\n(Authenticated)"]
```

```
    Auth --> Login["Login Screen"]
    Auth --> Register["Register Screen"]
    Auth --> ForgotPassword["Forgot Password"]

    App --> BottomTabs["Bottom Tab Navigator"]
    App --> Modal["Modal Stack"]

    BottomTabs --> Home["Home Tab\n(Stack)"]
    BottomTabs --> Explore["Explore Tab\n(Stack)"]
    BottomTabs --> Profile["Profile Tab\n(Stack)"]
    BottomTabs --> Settings["Settings Tab\n(Stack)"]

    Home --> HomeScreen["Home Screen"]
    Home --> DetailScreen["Detail Screen"]

    Modal --> ImageViewer["Image Viewer"]
    Modal --> ShareSheet["Share Sheet"]
```

Navigation library: `{{React Navigation v7 | Expo Router v4}}`

Deep link handling:

- URL scheme: `{{appname://}}`
- Universal links: `{{https://app.domain.com}}`
- See `push-notification-design.md` for notification deep links

Auth flow: Root navigator listens to auth state — no manual navigation required on login/logout.

---

4. Platform-Specific Considerations

Concern	iOS	Android	Solution
Back gesture	Swipe from edge	Back button + gesture	`hardwareBackPress` handler
Status bar	Overlaps content	Separate space	`SafeAreaView` + `StatusBar`
Permissions model	Request at use time	Request at use time + Manifest	`react-native-permissions`
Push notifications	APNs	FCM	Abstraction layer
Keyboard behavior	Push up content	May or may not push	`KeyboardAvoidingView`
Font rendering	System fonts crisp	Sub-pixel differences	Custom font loading

Concern	iOS	Android	Solution
Haptics	`UIFeedbackGenerator`	`Vibrator` API	`expo-haptics`
Secure storage	Keychain	Keystore	`expo-secure-store`

TODO: Add platform differences discovered during development.

---

5. Build Variants & Flavors

Variant	Bundle ID	API URL	Debug	Analytics	Push Env
Dev	`{{com.company.app.dev}}`	`http://localhost:4000`	Yes	Off	Development
Staging	`{{com.company.app.staging}}`	`https://api-staging.domain.com`	No	Off	Development
Production	`{{com.company.app}}`	`https://api.domain.com`	No	Yes	Production

Environment variable handling: `{{expo-constants | react-native-config}}`

TODO: Link to `.env.example` for each variant.

---

6. Code Sharing Strategy

```
shared/           # 80% of codebase — platform-agnostic logic
  ├── hooks/      # Custom hooks
  ├── services/   # API, storage abstractions
  ├── stores/     # State management
  └── utils/      # Pure utilities

platform/
  ├── ios/        # iOS-specific implementations
  └── android/    # Android-specific implementations

components/
  ├── Button.tsx          # Shared component
  ├── Button.ios.tsx      # iOS-specific override (if needed)
  └── Button.android.tsx  # Android-specific override (if needed)
```

Platform file resolution: Bundler automatically resolves `.ios.tsx` / `.android.tsx` before `.tsx`.

---

7. Native Module Integration

Module	Purpose	Source	Platform
`expo-camera`	QR scanning, photo capture	Expo SDK	Both
`expo-location`	Geolocation	Expo SDK	Both
`expo-notifications`	Push notifications	Expo SDK	Both
`expo-secure-store`	Keychain/Keystore	Expo SDK	Both
`expo-biometrics`	Face ID / fingerprint	Expo SDK	Both
`{{custom-native-module}}`	`{{PURPOSE}}`	Custom	`{{iOS/Android/Both}}`

New native module process:

1. Evaluate if Expo SDK covers the need
2. Check community modules (well-maintained, typed)
3. Write custom module as last resort
4. Native module must have TypeScript wrapper with full types

---

8. Performance Optimization Strategy

Strategy	Implementation	Status
JS thread optimization	Move heavy computation to worklets (Reanimated)	`{{Done/Planned}}`
Image caching	`expo-image` with disk cache	`{{Done/Planned}}`
List performance	`FlashList` instead of `FlatList`	`{{Done/Planned}}`
Bundle size	Hermes engine enabled, tree shaking	`{{Done/Planned}}`
JS startup time	Lazy loading of non-critical screens	`{{Done/Planned}}`
Memory management	Subscription cleanup in `useEffect`	`{{Done/Planned}}`
Render optimization	`React.memo`, `useCallback` on list items	`{{Done/Planned}}`

Performance targets:

- App cold start to interactive: < 2 seconds
- Screen transition: < 300ms
- List scroll: 60fps (120fps on ProMotion)

- Bundle size: < 20MB (iOS), < 15MB (Android)

---

9. Crash Reporting & Analytics Integration

Service	Purpose	Library
`{{Sentry}}`	Crash reporting, error tracking	`@sentry/react-native`
`{{Firebase Analytics}}`	User analytics, funnel tracking	`@react-native-firebase/analytics`
`{{Datadog}}`	APM, performance monitoring	`@datadog/mobile-react-native`

Privacy rules:

- No PII in analytics events
- User ID: hashed, not raw
- Comply with GDPR — analytics requires consent
- Crash reports: scrub sensitive data from breadcrumbs

Event naming convention: `{screen}_{action}` — e.g., `checkout_payment_started`

---

10. CI/CD for Mobile

```
flowchart LR
    PR["Pull Request"] --> UnitTests["Unit Tests\n(Jest)"]
    UnitTests --> E2ETests["E2E Tests\n(Detox / Maestro)"]
    E2ETests --> Build["Build\n(EAS Build / Fastlane)"]
    Build --> Distribute["Distribute\n(TestFlight / Firebase App Distrib)"]
    Distribute --> QA["QA Approval"]
    QA --> Store["Store Submission\n(EAS Submit / Fastlane deliver)"]
```

Stage	Tool	Trigger
Build	`{{EAS Build / Fastlane}}`	Push to `main` or `release/*`
Test distribution	`{{TestFlight / Firebase App Distribution}}`	Every staging build
E2E tests	`{{Detox / Maestro}}`	PR checks
App Store submit	`{{EAS Submit / Fastlane deliver}}`	Manual trigger (release manager)

Stage	Tool	Trigger
Code signing	`{{Expo managed / Fastlane match}}`	Automated

TODO: Link to CI configuration files in `.github/workflows/` or Fastfile.

---

11. Architecture Diagram

```
graph TB
    subgraph "Mobile App"
        UI["Screens & Components"]
        Nav["Navigation Layer"]
        State["State Management"]
        Services["Service Layer"]
        Native["Native Modules"]
    end

    subgraph "External"
        API["REST API"]
        Auth["Auth Service"]
        Push["Push Service\n(APNs / FCM)"]
        Analytics["Analytics\n(Firebase / Sentry)"]
    end

    UI --> Nav
    UI --> State
    State --> Services
    Services --> API
    Services --> Auth
    Native --> Push
    Services --> Analytics
```

Approval

Role	Name	Date	Signature
Author			
Mobile Lead			

Role	Name	Date	Signature
Tech Lead			
Product Owner			