# TapToPay with HCE Wallet & mPOS v1.0 Release Notes

This document outlines Version 1.0 of our Tap-to-Pay system(with a future HCE wallet and mPOS prospect), which successfully delivers end-to-end card-based user registration, secure email/password login, dynamic user profile retrieval, and peer-to-peer NFC-enabled fund transfers. It marks the completion of our core account lifecycle and implementation of a modular tap-to-pay system.

**Sign Up Page:**
Users go through a guided sign-up process, entering basic details before completing registration via NFC card tap. The app reliably reads physical NFC cards and extracts their UID, which is then associated with the user account during registration. The flow is intuitive, fully functional, and backed by a clean API call to the backend that adds the user data + uid to the database (model: User).
**Log In Page:**
Returning users can access their accounts using a standard email and password login form. Upon submitting their credentials, the app sends a POST request to the backend to authenticate the user against stored records (model: User). If valid, the user is logged in and routed to the profile interface.
**Profile Page:**
After login, users are directed to the Profile screen. This screen makes a POST request to the backend with the corresponding email and retrieves the most current data associated with the account. The result is rendered in a clean, responsive layout, providing users with a snapshot of their key information — including name, email, phone number, dob, password, status, balance, and associated cards. This ensures users always have access to real-time profile data. AND They can even edit all the details.
**Tap to Transfer (deprecated Merchant):**
In this legacy version of the merchant flow, the phone holder acts as the recipient. They input a desired amount, and the payer taps their NFC card to approve the transaction. The app reads the UID of the tapping card, fetches that user's account details from the backend, and deducts the specified amount from their balance.

---

**Navigation Architecture:**
The app features a clean, sidebar-driven navigation structure that organizes primary screens including Profile, Tap to Transfer (deprecated), and placeholders for upcoming features. The sidebar also includes a clearly marked Logout button, which securely returns users to the Login screen. On first launch, users land on the Login page with access to Sign-Up. Once logged in, the Profile screen serves as the default home landing page.
**Session Persistence:**
The app supports session persistence so that returning users who are already authenticated bypass the Login screen and are taken directly to their Profile home page. This creates a seamless experience by maintaining login state across app restarts, while still requiring login for unauthenticated users.
**UI Styling & Theming:**
The UI is modest with customized colors and typography, creating a polished visual identity that enhances usability with a basic design.

---

## Making the App Work Out-of-the-Box

The app is distributed as an **APK** installable on any Android device. Upon launch, it connects to a backend exposed via an **Ngrok tunnel**—a temporary, secure URL forwarding requests to a local development server. This enables full interaction with **user registration, login, profile, and fund transfer** features

## Current Deployment Stack Overview

The app's client side is built in **Android Studio** using **Kotlin** and **Jetpack Compose** for a modern, declarative UI approach. The app connects to a backend developed with **FastAPI**, a Python web framework designed for speed and simplicity. This backend is exposed to the internet via an **Ngrok subdomain**, forwarding external requests securely to my local development machine. For data persistence, the backend interacts with a **PostgreSQL database** deployed inside a **Docker container**, providing a reliable, containerized environment to manage user and transaction data. T

## Project Sharing

The project will be hosted on **GitHub** to provide full access to the codebase, version control, and downloadable APK releases. Additionally, detailed write-ups documenting the development process will be published on Medium (link) and Dev.to (link).

---

## Future Expansion & File Structure Overview

The next major milestones for the app include adding a **mobile point-of-sale (mPOS) flow** for merchants and a **HCE-Based Attendance system.**

### 🔧Backend (Python):

- main.py (FastAPI server logic): will handle new API routes for mPOS and card emulation.
- models.py: will expand to include additional entities (e.g., transactions, emulated cards).
- database.py: may require adjustments to connection logic or can be extended into additional DB-layer modules for clarity.

### 📱 Android (Kotlin):

- MainActivity.kt: will evolve to manage navigation or shared state across new screens.
- MerchantActivity.kt (Tap to Transfer): will likely be refactored or replaced by the mPOS flow.
- SignUpActivity.kt and LoginActivity.kt: likely remain stable with minor validation or UI polish.
- SignUpScreen, ProfileScreen, LoggedInHomeScreen:
    - ProfileScreen will expand to show new features (e.g., linked emulated cards).
    - LoggedInHomeScreen will route users to new feature screens.
    - New Composable screens will be added alongside existing ones to support mPOS and emulator logic.