1. Introduction

This document outlines the Proof of Concept (POC) for integrating UPI (Unified Payments Interface) as a replacement for Stripe in Wolonote's external payment processing system. The POC demonstrates the ability to:

- Top-Up User Balance via UPI Collect Requests.
- Process Withdrawals via UPI PayOut.
- Authenticate and execute transactions securely using Razorpay's UPI API.

2. Objective of the POC

Currently, Stripe is used for:

- Top-Up: Adding funds from external sources into Wolonote.
- Withdrawal: Transferring funds from Wolonote to an external bank account.

Since Stripe does not support UPI, we are replacing it with UPI APIs to handle these transactions specifically for India.

3. Why Razorpay Over Other UPI Providers?

While evaluating UPI payment providers, we considered **Razorpay, Paytm, Cashfree, and PhonePe**. The decision to use **Razorpay** was based on the following factors:

Feature	**Razorpay**	**Paytm**	**Cashfree**	**PhonePe**
UPI Collect (Top-Up)	Yes	Yes	Yes	Yes
UPI PayOut (Withdrawals)	Yes	Yes	Yes	Yes
API Documentation & of Use Ease	Excellent	Limited	Good	Limited
Sandbox Environment	Available	No	Available	No
Security & Compliance	Strong	Strong	Strong	Strong
Support for Scaling	High	Medium	High	Medium

Key Reasons for Choosing Razorpay:

→ **Best API Documentation** – Razorpay offers clear and well-structured API documentation, making integration easier.

→ **Sandbox Testing Support** – Unlike Paytm and PhonePe, Razorpay provides a test environment, allowing us to validate transactions before going live.

→**Reliable UPI PayOut API** – Razorpay offers stable & scalable PayOut APIs with minimal downtime.

→ **Industry Adoption** – Many Indian startups and businesses use Razorpay, ensuring a trusted payment ecosystem.

→ **Ease of Authentication** – Razorpay supports API key-based authentication, making secure transactions easier.

4. API Endpoints Implemented

I have used placeholder API keys for the implementation. These will be replaced with the actual Razorpay API keys once they are received.

→ UPI Collect (Top-Up)

- Endpoint: `GET /upi/collect?amount={amount}&user_id={user_id}`
- Description: Generates a UPI payment request link for the user to initiate a top-up.

→ UPI PayOut (Withdrawal)

- Endpoint: `GET/upi/payout?amount={amount}&user_id={user_id}&upi_id={upi_id}`
- Description: Initiates a withdrawal request from Wolonote to a user's UPI ID.

5. RazorpayX & Payout Process

RazorpayX is used for **processing payouts**, and the payout function performs the following key actions:

Step 1: Create a Contact in RazorpayX

- A **contact** represents the recipient who will receive the payout.
- The API call includes the recipient's name, email, and phone number.
- Once created, **RazorpayX returns a `contact_id`**, which is required for the next step.

Step 2: Create a Fund Account for UPI

- A **fund account** links the recipient's UPI ID to the **RazorpayX contact**.
- The API call includes the `contact_id` and UPI ID (`upi_id`).
- Once created, **RazorpayX returns a `fund_account_id`**.

Step 3: Initiate a Payout

- The payout request is initiated using:
    - `account_number`: RazorpayX Business Account.
    - `fund_account_id`: Retrieved from Step 2.
    - `amount`: Converted to **paise** before processing.
    - `mode`: UPI.
    - `purpose`: Refund (or any valid reason).
- If successful, RazorpayX marks the payout **processed**.
- In **test mode**, the system **automatically returns `processed` status** to ensure proper flow

6. Implementation Overview

→ Secure API Key Storage

I have stored the API keys in the `.env` file for security purposes, ensuring they are not exposed or accessible by others.

→ Go Code Overview

- **Backend Framework**: Go (Golang) with the `Echo` web framework.
- **Top-Up Handling (`upiCollect`)**:
    - Sends a **UPI Collect request** to Razorpay.
    - Stores metadata (user ID) in the request.
- **Payout Handling (`upiPayout`)**:
    - Creates a **contact** and **fund account**.

    - Initiates a **payout request** using RazorpayX.
    - In test mode, always returns a **successful payout response**.
- **Authentication**:
    - API requests are authenticated using **Basic Authentication (API Key & Secret)**.
- **Error Handling & Logging**:
    - Logs API responses and errors for debugging.
    - Returns meaningful error messages to the client.