1. Overview

Product name: Nuvei REST API 1.0 Emulator
Owner: Solutions & Implementation
Stakeholders: Solutions Engineering, Product, Integrations/PS, Merchant Devs, Sales Engineering

1.1 Purpose

Provide a **simple, guided, browser-based emulator** for the Nuvei **REST API 1.0 (PPP)** that:

- Uses **real Nuvei sandbox endpoints**, not mocks
- Feels like a **product**, not a developer tool
- Wraps our existing Postman collection into **1-click scenarios**
- Makes it easy for **non-Postman** users to:
    - Run full 3DS flows (challenge, frictionless, MPI-only, external MPI)
    - Run basic auth/capture, recurring, payouts
    - See exactly what is sent and received

1.2 Problem statement

Today:

- Most flows are only accessible via **Postman**:
    - Requires understanding pre-request scripts and environment variables
    - Hard to "tell a story" to merchants and internal stakeholders
- Non-technical users find Postman **intimidating** and confusing
- Even technical users waste time **chaining multiple calls manually**

We need a **Nuvei-branded emulator** that pre-packages these flows into scenarios such as:

- "3DS Challenge – Auth + Liability Shift"
- "3DS Frictionless"

- “Non-3DS Auth + Settle”
- “External MPI payment”
- “Recurring with 3DS”
- “Payout demo”

2. Scope

2.1 In scope (v1)

- **APIs (REST API 1.0 / PPP)**:
  - `getSessionToken`
  - `initPayment`
  - `payment` (non-3DS, 3DS challenge, frictionless)
  - `authorize3d` / `verify3d` (MPI only)
  - `settleTransaction`, `voidTransaction`
  - `getPaymentStatus`
  - `payout` (if available and useful in the collection)
- **Flows / scenarios**:
  - 3DS challenge & frictionless
  - Non-3DS auth + capture
  - External MPI
  - MPI-only (3DS as a service)
  - Recurring
  - Payout
  - Negative paths: errors, invalid checksum, declines
- **Environments**:

    - Nuvei **sandbox only** (PPP test endpoint)
    - Merchant config via UI (merchant ID, site ID, secret, URL)
- **UI**:
    - Web front-end (SPA) with:
        - Sandbox credential form
        - Scenario catalog (cards)
        - Scenario detail + stepper
        - JSON request/response viewers
        - 3DS challenge iframe

2.2 Out of scope (v1)

- Production credentials / live traffic
- Full user management, roles & permissions
- Code generators for each language (could be v2)
- Observability / per-merchant analytics (tie-in to MCP is v2+)

3. Personas & use cases

3.1 Personas

- **Solutions Engineer (Primary)**
    - Needs to run live flows during calls in <5 minutes
    - Wants reliable demo cards for “3DS challenge”, “payout”, etc.
- **Merchant Developer (Technical / Semi-technical)**
    - Understands HTTP but doesn’t want to debug checksum logic
    - Wants to copy-paste JSON payloads into their own stack
- **Product / Sales / PM**

    ◦ Wants a safe, visual explanation of how Nuvei behaves in real flows

3.2 Main use cases

1. **UC1 – First-time sandbox validation**
    ◦ Input sandbox credentials
    ◦ Run “Smoke test: getSessionToken”
    ◦ Confirm environment is working
2. **UC2 – Explain 3DS challenge flow**
    ◦ Run “3DS Challenge – Auth + Liability Shift”
    ◦ Show each step and 3DS iframe
    ◦ Share run as HTML/PDF with merchant
3. **UC3 – Compare 3DS vs non-3DS UX**
    ◦ Run “Non-3DS Auth + Settle”
    ◦ Run “3DS Frictionless”
    ◦ Show difference in steps & responses
4. **UC4 – Show external MPI integration**
    ◦ Run “External MPI – Liability Shift”
    ◦ Highlight `eci`, `cavv`, `dsTransID` values
5. **UC5 – Debug errors**
    ◦ Run “Failure – Invalid checksum” and “Failure – Decline”
    ◦ Help developers recognise typical error structures

4. Functional requirements

4.1 Environment & credentials

- **FR-ENV-1 – Credential form**

    - Inputs:
        - Sandbox base URL
        - Merchant ID
        - Merchant Site ID
        - Secret key
        - Algorithm (SHA-256, SHA-1 – but default SHA-256)
    - Actions:
        - Validate format
        - “Test connection” button → calls `getSessionToken`
        - Show success/failure banner
- **FR-ENV-2 – Storage**
    - v1:
        - Credentials **not stored** on the server
        - Option to store locally in browser (encrypted) with warning
- **FR-ENV-3 – Multiple environment profiles (optional v1.1)**
    - Allow saving named profiles: “Demo merchant”, “Merchant X sandbox”

4.2 Scenario catalog & navigation

- **FR-SCEN-1 – Scenario gallery**
    - Grid/list of scenario cards
    - Each card:
        - Name
        - Short description

            - Tags (3DS, CARD, PAYOUT, ERROR, RECURRING, MPI)
    - Filters: `3DS`, `CARD`, `PAYOUT`, `ERROR`, etc.
- **FR-SCEN-2 – Scenario detail**
    - Layout:
        - Left: stepper (each step = specific API call)
        - Right: selected step details:
            - Summary (NL explanation)
            - Request & response JSON (collapsible)
            - Status: Pending / Success / Error
- **FR-SCEN-3 – Execution modes**
    - "Run full scenario"
    - "Step-by-step" mode
    - Ability to re-run:
        - Entire scenario
        - Individual step, keeping context

4.3 Step execution & context

- **FR-STEP-1 – Context object**
    - Per run, the emulator maintains a **context**:
        - Env: merchant ID, site ID, secret, base URL
        - Runtime:
            - `sessionToken`
            - `transactionId`, `relatedTransactionId`
            - 3DS-related: `acsUrl`, `cReq`, `threeDVersion`

        - Recurring tokens, payout IDs

- **FR-STEP-2 – Request templates**
    - Each step has a request template with placeholders:
        - `{{env.merchantId}}`, `{{ctx.sessionToken}}`, `{{meta.timestamp}}`, etc.
    - Before sending:
        - Resolve placeholders from env + context
        - Calculate checksum and inject
- **FR-STEP-3 – Checksum logic**
    - For each endpoint, define `checksumFields`:
        - Example: `getSessionToken` uses `[merchantId, merchantSiteId, clientRequestId, timeStamp]`
        - Example: `payment` uses `[merchantId, merchantSiteId, clientRequestId, amount, currency, timeStamp]`
    - Concatenate fields + secret key in correct order
    - Hash with selected algorithm
    - Automatically update when user edits relevant fields

4.4 3DS & MPI behaviour

- **FR-3DS-1 – 3DS challenge**
    - When response contains `acsUrl` + `cReq` (or equivalent):
        - Show “3DS challenge required” status
        - Open iframe/modal pointing to Nuvei 3DS simulator URL
        - After user completes challenge, resume next step
- **FR-3DS-2 – 3DS frictionless**

    - When response indicates frictionless:
        - No iframe
        - Highlight 3DS fields (version, eci) in response panel
- **FR-3DS-3 – MPI-only flows**
    - Steps:
        - `authorize3d`
        - 3DS simulator (if challenge)
        - `verify3d`
    - Highlight values merchant must consume:
        - `eci`, `cavv`, `dsTransID` (or equivalent)

4.5 Logging, export & sharing

- **FR-LOG-1 – Local run log**
    - Minimum:
        - Scenario ID
        - Timestamp
        - Step results (success/error)
        - Transaction IDs
    - Stored in browser only for v1
- **FR-LOG-2 – Export**
    - Export current run as:
        - JSON (sanitised – no PAN, no secret)
        - HTML/PDF summary (for sharing with merchants)
- **FR-LOG-3 – Shareable link (v1.1+)**
    - Generate URL that contains:
        - Scenario ID

        - Custom parameters (amount, currency, tags)
    - **Never** include credentials in the link

5. UX & UI requirements

5.1 General

- Responsive, single-page layout
- Nuvei-branded (colors, fonts)
- Clear separation:
    - Config (env)
    - Scenario selection
    - Scenario execution & details

5.2 Key UX decisions

- For **non-technical users**, show a human summary first, JSON second:
    - "We sent an authorization for 150 EUR on test Visa and received APPROVED."
- Clear color code:
    - Green: approved / success
    - Orange: redirect / challenge
    - Red: errors / declines
- Show "what matters":
    - Step result
    - Transaction IDs
    - 3DS indicators
    - Error codes

6. Technical architecture

6.1 High-level

- **Front-end**
  - React / Next.js SPA
  - Talks only to emulator backend, not directly to Nuvei (except for 3DS iframe)
- **Backend**
  - Lightweight Node.js / NestJS or similar
  - Stateless, suitable for serverless (Cloudflare Workers, AWS Lambda, etc.)
  - Responsibilities:
    - Scenario engine
    - Context handling
    - Checksum calculation
    - Proxying to Nuvei sandbox endpoints
    - Response normalisation

6.2 Emulator backend API (internal)

- `POST /api/env/test`
  - Body: env config (merchant ID, site ID, secret, URL, algorithm)
  - Action: run `getSessionToken` and return status
- `GET /api/scenarios`
  - Returns: list of scenario metadata
- `GET /api/scenarios/:id`
  - Returns: full scenario definition
- `POST /api/scenarios/:id/run`

    - Body: `{ envConfig, overrides? }`
    - Executes all steps, returns run summary + step details
- `POST /api/scenarios/:id/step/:stepId/run`
    - Runs a single step with existing context

6.3 Scenario model

- See JSON config section below for exact example

7. Security & privacy

Even for sandbox, treat this carefully.

- **No secrets in code or repo**
    - Merchant IDs, site IDs, secret keys must **never** be committed
    - All secrets provided at runtime
- **Data masking**
    - PAN masked to last-4 in UI and logs
    - Secret key never displayed back to the user after input
- **Transport**
    - HTTPS only for emulator and sandbox
- **Rate limiting**
    - Basic per-IP rate limit to avoid sandbox abuse

8. Non-functional requirements

- **Performance**
    - Emulator overhead < 200 ms per step
- **Availability**
    - Deployed on resilient serverless / container platform

- **Extensibility**
    - New scenarios added by editing JSON config, not code where possible
- **Observability (later)**
    - Log basic metrics (count of runs, scenario IDs, error rates), *without* storing secrets

9. Roadmap

- **v0.1 – MVP**
    - 1–2 scenarios:
        - 3DS Challenge – Auth + Liability Shift
        - Non-3DS Auth + Settle
    - Manual scenario definitions (hardcoded)
- **v0.2 – Scenario config**
    - Implement scenario JSON config loader
    - Add 5+ core scenarios
    - Stable checksum & context handling
- **v0.3 – 3DS / MPI / Recurring**
    - Add 3DS frictionless, MPI-only, recurring, payout
- **v1.0 – Polish & documentation**
    -