Setula Hackathon Submission

Setula — Hackathon Submission

Project title

Setula

Tagline

Cross-border contractor payments with USDC settlement on Arc.

Description

Setula helps UAE agencies pay overseas contractors in local currency, with USDC settling on Arc under the hood. The finance user sees only the AED amount they approve and the INR amount the contractor receives. Underneath, Circle developer-controlled wallets execute a real USDC transfer on Arc Testnet, producing a publicly verifiable transaction hash on ArcScan. The payment is tracked through seven states — from quote to delivery — with duplicate prevention, failure isolation, and an invoice-linked receipt that ties every reference together.

Hackathon scope: One UAE agency, one India-based contractor, one invoice (INR 91,000), a sandbox AED-to-INR quote (22.75 INR/AED), simulated AED funding, a real 0.01 USDC Arc Testnet settlement, simulated INR bank payout, and an invoice-linked receipt with ArcScan evidence.

Track

Payments

Circle account email

nedupowei22@gmail.com

Circle products used

- **Circle Developer-Controlled Wallets** (SDK v10.8) — manages sender and recipient EOAs on Arc Testnet, executes USDC transfers, polls transaction status until `COMPLETE`
- **Arc Testnet** — settlement network for real USDC transfers, produces publicly verifiable transaction hashes via ArcScan

Other products / infrastructure

Product	Purpose
Circle Developer-Controlled Wallets	Wallet management and USDC transfer execution
Arc Testnet	Settlement network
USDC	Settlement asset
ArcScan	Public transaction explorer and evidence
Vercel	Frontend hosting (Next.js landing page + docs)
Railway	Backend hosting (Node.js HTTP server)

Working MVP

- **Live application:** https://setula.vercel.app
- **Source code:** https://github.com/OutstandingVick/setula
- **Documentation:** https://setula.vercel.app/docs/overview
- **ArcScan proof:** https://testnet.arcscan.app/tx/0xd033f753ccae0b55585a94a754657a6154f2a6690748b55adbdbb469e2b2afec
- **Backend health:** https://ideal-alignment-production-912d.up.railway.app/health

How to test the MVP

1. Open https://setula.vercel.app — the landing page loads with a contractor invoice for INR 91,000.
2. Click **Launch demo** — the payment page loads with the pre-filled sandbox quote.
3. Click **Approve payment** — simulated AED funding, then a real 0.01 USDC transfer executes on Arc Testnet.
4. Watch settlement progress through `SETTLEMENT_PENDING` → `SETTLED`.
5. Click the ArcScan link to view the confirmed on-chain transaction.
6. Click **Confirm delivery** — the simulated INR payout callback fires.
7. View the receipt with all linked references and ArcScan evidence.

Validated states

```
DRAFT → QUOTED → FUNDED → SETTLEMENT_PENDING → SETTLED → PAYOUT_PENDING → DELIVERED
```

Edge cases tested

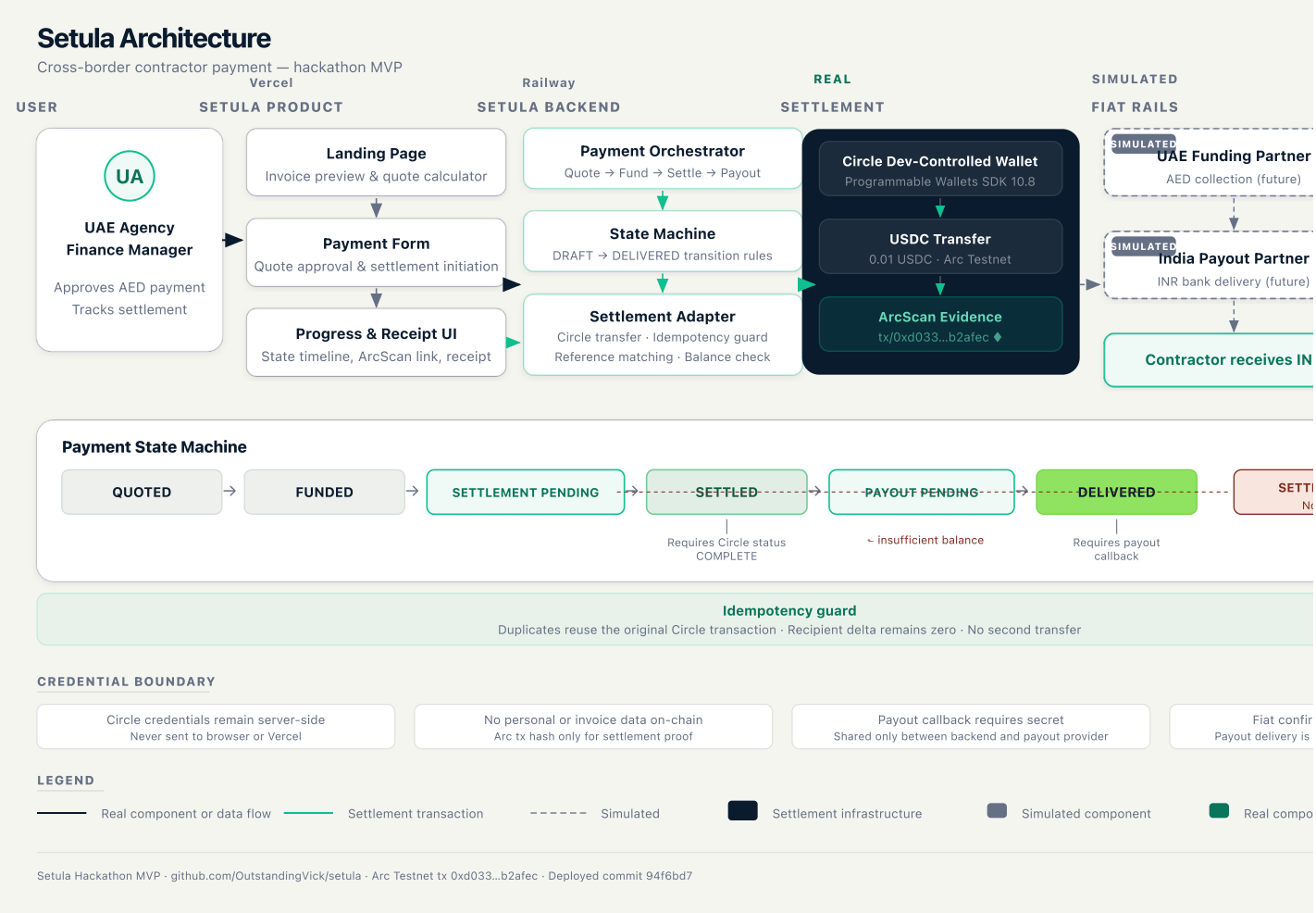
- Duplicate settlement prevention (recipient delta 0)
- Insufficient-balance failure (`SETTLEMENT_FAILED`, no ArcScan link)
- Invalid payout callback rejection
- Page refresh during progress
- Mobile (390px) and desktop (1440px) responsive QA

Validation record

Five consecutive deployed runs completed. Circle transaction status `COMPLETE`. ArcScan HTTP 200. All payments reached `DELIVERED`. Secret scan passed — no credentials in tracked files or browser assets.

Full validation: STABLE_DEPLOYMENT_VALIDATION.md

Architecture diagram



Setula Architecture Diagram

The architecture has five columns: User → Setula Frontend (Vercel) → Setula Backend (Railway) → Settlement (Circle / Arc Testnet) → Fiat Rails (simulated). The payment state machine sits beneath with the full `QUOTED → DELIVERED` progression, including the `SETTLEMENT_FAILED` branch and idempotency guard.

Video demo

https://drive.google.com/drive/folders/1bnqxXauob-Wh5iEtqh0iGIMOQfuGNnHh?usp=sharing

Documentation

- **README:** https://github.com/OutstandingVick/setula
- **Docs site:** https://setula.vercel.app/docs/overview
- **API & State Machine:** https://setula.vercel.app/docs/api
- **Architecture:** https://setula.vercel.app/docs/architecture
- **Quickstart:** https://setula.vercel.app/docs/quickstart
- **Deployment:** https://setula.vercel.app/docs/deployment
- **Local Development:** https://setula.vercel.app/docs/local-development

What is real and simulated

Component	Status
Quote generation	Sandbox (fixed rate)
AED collection	Simulated
Circle wallet settlement	**Real**
USDC transfer on Arc Testnet	**Real**
ArcScan transaction proof	**Real**
INR bank payout	Simulated
Invoice/receipt reconciliation	**Real** (application logic)

Product feedback

What worked well

- **Circle Developer-Controlled Wallets SDK** was straightforward to integrate. The `createTransfer` and `getTransfer` APIs provided clear transaction lifecycle tracking. The SDK's TypeScript types aligned well with Setula's Zod-validated domain model.
- **Arc Testnet** provided fast, deterministic settlement with publicly verifiable transaction hashes on ArcScan. This gave us machine-verifiable proof without needing a custom block explorer.
- **USDC as the settlement asset** kept the value leg simple — no volatile token prices to reconcile against fiat invoice amounts.

Challenges

- **Transaction polling** required implementing a retry loop with configurable intervals and timeouts. A webhook or event-stream mechanism for settlement confirmation would simplify backend state management.
- **Balance checking** before transfer submission is critical — the SDK returns a clear error for insufficient funds, but pre-flight balance validation avoids a wasted transfer attempt.
- **ArcScan URL construction** from the transaction hash was manual. A first-class SDK utility to generate explorer URLs would reduce boilerplate.

Suggestions for Circle / Arc

- A **webhook or streaming event** system for transfer status changes would eliminate polling and reduce latency in payment UIs.
- **Sandbox/testnet wallet funding** via a faucet or API would accelerate developer onboarding — manually funding wallets for every test run added friction.
- **ArcScan API** for programmatic transaction verification (status, confirmations, timestamp) would complement the block explorer and enable automated receipt generation.