

SteelLink: AI + Blockchain Enabled B2B Vendor Platform

Pramod Dhase¹, Niraj Chaudhari¹, Samruddhi Dorge¹, Sapna Tandel¹, Prof. N.H.Despande²

¹ Students, Department of Information Technology Sinhgad College of Engineering, Vadgaon(BK), Pune

² Assistant Professor, Department of Information Technology

Abstract - The Indian steel industry, one of the largest in the world, continues to suffer from systemic inefficiencies rooted in fragmented vendor ecosystems, lack of trust in documentation, opaque pricing, and material wastage from un-sold leftover stock. Traditional procurement processes rely heavily on personal contacts, phone calls, and unverified paper documents, creating significant barriers for smaller buyers and vendors trying to participate in national and international supply chains.

This paper presents *SteelLink*, a full-stack, AI + Blockchain enabled Business-to-Business (B2B) procurement platform designed to modernize the steel procurement lifecycle. The system introduces a verified vendor registry anchored to the Ethereum blockchain using Solidity smart contracts, an AI-driven anomaly detection engine using scikit-learn's Isolation Forest algorithm (trained live on Supabase pricing data) to flag suspicious listings, and a structured Request for Quote (RFQ) bidding system allowing buyers to post procurement requirements and receive competitive bids from verified vendors. The platform is built on a modern technology stack comprising Next.js (Frontend), Node.js/Express (Backend), Supabase (PostgreSQL-based BaaS), and a Python AI monitoring daemon.

Experimental validation confirms that the AI engine correctly flags price anomalies using Isolation Forest (scikit-learn), while the blockchain service provides immutable on-chain verification of vendor documents via Ethereum smart contracts. The platform is deployed locally with a scalable cloud-ready

architecture designed for future Vercel/Render deployment.

Keywords— Blockchain, Smart Contract, Ethereum, B2B, Procurement, AI, Anomaly Detection, Supabase, Next.js, Steel Industry, RFQ, Vendor Verification.

1. INTRODUCTION

India is the second-largest producer of steel globally, contributing over 125 million metric tonnes (MT) of crude steel annually [1]. Despite this massive production capacity, the steel procurement ecosystem is rife with structural inefficiencies. Buyers ranging from large infrastructure contractors to small fabricators must navigate a fragmented marketplace where vendor credibility is unverifiable, pricing is opaque, and leftover material stock sits idle at factory yards due to lack of demand visibility.

The problem is multi-dimensional. Firstly, **vendor fraud** is a persistent challenge. Counterfeit quality certificates (e.g., fake IS 2062 grade certifications) inflate material costs and compromise structural integrity in construction projects. Secondly, **information asymmetry** between buyers and vendors leads to inefficient price discovery, with smaller buyers consistently paying premiums above market rate. Thirdly, significant quantities of **leftover stock** material cut-offs and surplus inventory from large-scale projects are never efficiently redistributed, leading to quantifiable financial loss across the supply chain.

Existing B2B steel marketplaces in India (e.g., Steel360, Mandi Gobindgarh networks) operate as classified advertising boards without trust infrastructure, pricing intelligence, or structured RFQ mechanisms. Platforms popular in Western markets

(e.g., Xometry) are optimized for precision manufacturing and do not cater to the bulk, commodity-grade steel specifications common in Indian procurement.

SteelLink is designed to address this gap. It is not a payment platform all financial transactions between buyers and vendors are agreed upon and settled offline through standard commercial instruments (UPI, bank transfer, cheque). SteelLink exclusively manages the *procurement lifecycle*: discovery, verification, RFQ, and bidding. This deliberate scoping ensures regulatory simplicity while focusing the platform's core value on trust and efficiency.

The primary contributions of this work are:

1. **Blockchain Vendor Registry:** An immutable on-chain record of verified vendor documents using Ethereum smart contracts, making fraudulent certificates technically unsustainable.
2. **AI Price Intelligence Engine:** A statistical AI microservice that analyses listing prices against market baselines to flag anomalies in real time.
3. **Structured RFQ System:** A formal bidding workflow replacing informal phone-based procurement with a structured, auditable digital process.
4. **Leftover Material Exchange:** A dedicated marketplace feed for buyers to discover and acquire off-cut and surplus steel stock.

2. LITERATURE SURVEY

2.1 Blockchain in Supply Chain and Procurement

The application of blockchain technology to supply chain management has been an active research domain since Nakamoto's foundational paper on decentralised ledgers [2]. Kshetri [3] demonstrated that blockchain can provide unprecedented levels of supply chain transparency by creating an immutable audit trail of transactions and document hashes. In the context of manufacturing, Saberi et al. [4] highlighted that smart contracts can automate compliance checks and vendor verification without relying on centralised intermediaries, reducing both

costs and processing time.

In the specific context of Indian industrial procurement, blockchain adoption remains nascent. Most implementations are limited to pilot studies in pharmaceutical supply chains and agricultural commodity tracking (e.g., ITC's e-Choupal derivatives). The steel sector has received comparatively little academic attention despite its scale, representing a clear research gap that SteelLink seeks to address.

2.2 Artificial Intelligence in B2B Market Intelligence

Machine learning techniques have been applied extensively to B2B pricing and fraud detection. Chandola et al. [5] provide a comprehensive survey of anomaly detection methods, noting that statistical approaches (Z-score, IQR-based filtering) remain highly effective for univariate price data due to their interpretability and low computational overhead. Deep learning models such as Autoencoders [6] offer superior performance on high-dimensional datasets but introduce complexity that is disproportionate to the scale of an SME-oriented commodity platform.

For SteelLink's AI engine, the **Isolation Forest** algorithm (scikit-learn) was chosen: it is an unsupervised ensemble method that isolates anomalies by recursively partitioning the feature space. Unlike Z-score, it handles non-Gaussian price distributions gracefully and requires no labelled training data. The model is trained on a combination of synthetic baseline pricing data and live historical records fetched directly from the Supabase materials table, making it self-improving as the platform accumulates real transaction data.

2.3 B2B Procurement Platforms and RFQ Systems

Request-for-Quote (RFQ) systems are the standard mechanism for formal B2B procurement. Dai and Kauffman [7] analysed the efficiency gains from electronic RFQ systems, demonstrating an average 14% reduction in procurement costs and a 40% reduction in cycle time compared to manual processes. Modern platforms such as Ariba (SAP) and Coupa implement complex RFQ workflows but are designed for large enterprises with dedicated procurement teams, creating a capability gap for the

SME segment that dominates the Indian steel fabrication market.

2.4 Full-Stack Platform Architectures

The architectural decision to use Next.js with a Node.js backend and a separate Python AI microservice follows the Service-Oriented Architecture (SOA) pattern. Fowler [8] articulates the value of independently deployable services, noting that decoupling the AI inference engine from the core API server allows each component to be scaled independently based on load. The choice of Supabase as a Backend-as-a-Service (BaaS) layer provides an additional advantage: it combines PostgreSQL's relational capabilities with a built-in REST API and Row Level Security (RLS) policies, eliminating the need for a dedicated ORM layer in the standard CRUD operations.

SteelLink is optimized for high-speed, low-cost operations by targeting the **Polygon Amoy Testnet** as its primary blockchain host. Unlike Ethereum Sepolia, which can experience congestion and higher gas price fluctuation even on testnets, Polygon's Layer-2 architecture ensures near-instant transaction finality and consistent sub-cent gas costs, making it the superior choice for high-frequency industrial procurement events like vendor verification and order ledger anchoring.

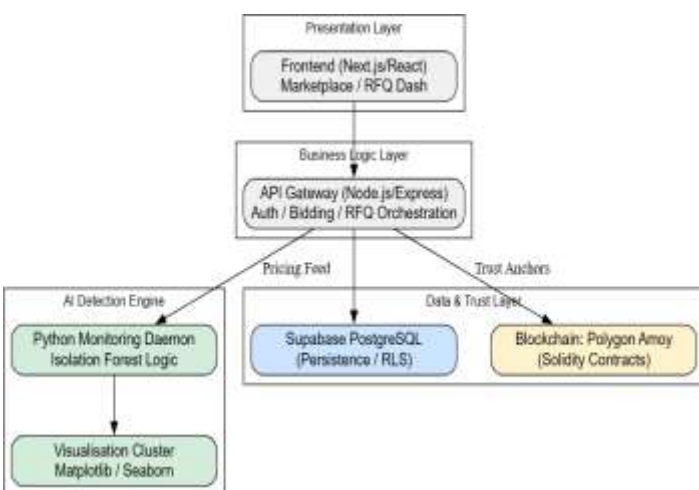


Figure 1: SteelLink High-Level System Architecture and Data Flow

Table 1: Comparative Analysis of Related B2B Procurement Platforms

Platform	Blockchain	AI Pricing	RFQSystem	Limitation
Steel360 (India)	None	None	Basic Listing	No trust infrastructure
Ariba (SAP)	None	Partial	Advanced	Enterprise only, costly
Xometry (USA)	None	Yes	Yes	Not India/steel focused
TradeBriefs	None	None	None	News aggregator only
SteelLink (Proposed)	Ethereum	Isolation Forest	Full Bidding	Blockchain gas cost

3. SYSTEM ARCHITECTURE

3.1 High-Level Architecture

SteelLink follows a layered, microservice-oriented architecture. As illustrated conceptually, the system comprises five distinct layers that communicate via REST APIs:

- Frontend Layer (Next.js):** Provides the user interface for Buyers, Vendors, and Admins. Built with React 19, TailwindCSS v4, and Lucide icons.
- API Gateway Layer (Node.js/Express):** Handles all business logic, authentication, and routing. Communicates with Supabase for data persistence and the Python AI engine for price analysis.
- Database Layer (Supabase/PostgreSQL):** Stores all relational data: user profiles, material list-ings, RFQs, and bids. Row Level Security (RLS) enforces data access policies.
- AI Microservice Layer (Python):** A standalone Python script (main.py) using scikit-learn's Iso-lation Forest model for price anomaly detection. It reads live pricing data

from Supabase and generates 2D/3D visualisation charts (Seaborn, Matplotlib, Plotly) exported to the frontend's public/directory. A Streamlit dashboard (streamlit_app.py) provides an admin-facing analytics inter-face.

5. **Blockchain Layer (Polygon Amoy/Hardhat):** Two Solidity smart contracts deployed on the Polygon Amoy Testnet handle immutable vendor verification and order ledger recording with high-speed finality.

3.2 Data Flow

The standard procurement data flow in SteelLink proceeds as follows. A **Vendor** registers on the platform, submits their business documents (GST certificate, BIS licence), and an **Admin** reviews the sub-mission. Upon approval, the admin triggers the verifyVendor() function on the deployed VendorVerification.sol contract. This function stores a SHA-256 hash of the vendor's documents on-chain, making subsequent forgery detectable through hash comparison. The vendor's profile is then updated in Supabase with the blockchain transaction hash as proof.

A **Buyer** visiting the platform can post an RFQ specifying material grade, quantity (MT), delivery location, and deadline. Verified vendors receive notifications and can submit structured bids with price-per-MT and delivery timeline. The AI engine monitors incoming material listings via a 5-second polling loop and flags statistical outliers (Isolation Forest score = -1) for admin review.

3.3 Database Schema

The Supabase PostgreSQL database comprises four core tables linked by foreign key relationships:

Table 2: SteelLink Core Database Schema (Supabase/PostgreSQL)

Table	Key Columns	Relationships
users	id (UUID), role, company_name, is_verified, verification_hash	Root entity

materials	id, name, quantity, price_per_unit, status, vendor_id, blockchain_hash	FK: users.id
rfqs	id, title, quantity, location, deadline, status, buyer_id	FK: users.id
bids	id, amount, proposal, delivery_time, status, rfq_id, vendor_id	FK: rfqs.id, users.id
orders	id, rfq_id, bid_id, buyer_id, material_name, total_price, blockchain_hash	FK: rfqs.id, bids.id

4. BLOCKCHAIN IMPLEMENTATION

4.1 Smart Contract Design

Two Solidity smart contracts, compiled with solidity 0.8.28 and deployed via Hardhat, form the trust backbone of SteelLink:

VendorVerification.sol maintains a mapping from vendor Ethereum wallet addresses to a Vendor struct containing the vendor's name, a SHA-256 document hash (bytes32), verification status, and timestamp. The verifyVendor() function is protected by an onlyAdmin modifier, ensuring only the platform's admin wallet can commit verifications. The usedHashes mapping prevents the same document hash from being re-submitted, directly countering the re-use of authentic documents to verify fraudulent entities. **OrderLedger.sol** provides an immutable record of procurement orders. Once a buyer accepts a bid offline, the admin can call recordOrder(bytes32 orderHash, string status) to anchor the order's SHA-256 hash on-chain. Subsequent calls to updateStatus() allow the order

status to progress from "confirmed" to "dispatched" to "delivered", creating a full on-chain audit trail.

4.2 Key Smart Contract Functions

Listing 1: Core VendorVerification.sol functions

```
function verifyVendor( address _v, string memory _n,
bytes32 _h) public onlyAdmin
{
    require(! usedHashes[_h], "Hash verified");
    vendors[_v] = Vendor({ name: _n, docHash: _h,
        isVerified: true, timestamp: block.
            timestamp});
    usedHashes[_h] = true;
    emit VendorVerified(_v, _n, _h);
}
```

4.3 Deployment Strategy

The contracts are compiled with **Solidity 0.8.28** (as declared in hardhat.config.js) and can be deployed primarily to the **Polygon Amoy Testnet** (--network amoy), which offers the lowest gas costs and fastest block times. The system also maintains compatibility with the **Ethereum Sepolia Testnet** and a **local Hardhat node** (--network localhost) for development. The Polygon Amoy network is configured with a public RPC (<https://rpc-amoy.polygon.technology>), facilitating easy on-chain verification without the bottleneck of high testnet gas prices frequent on Ethereum. Contract addresses are stored in backend/.env for backend-blockchain interaction via the Ethers.js v6 library.

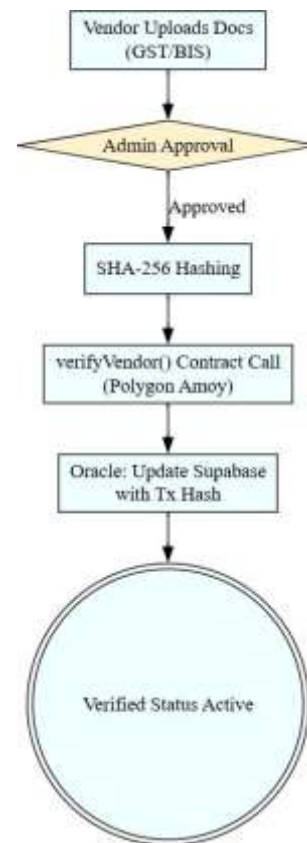


Figure 2: Immutable Vendor Verification and Order Ledger Workflow (Polygon Amoy)

Table 3: Blockchain Deployment Configuration

Parameter	Value
Supported Networks	Sepolia (Ethereum), Amoy (Polygon), Localhost
Solidity Version	0.8.28
Development Framework	Hardhat + @nomicfoundation/hardhat-toolbox
Contract 1	VendorVerification.sol
Contract 2	OrderLedger.sol
Gas Token	Test ETH / Test MATIC (No real-world value)
Backend Library	ethers.js v6.16.

5. AI ENGINE: PRICE ANAMOLY DETECTION

5.1 Anomaly Detection Methodology

The SteelLink AI Engine (ai-engine/main.py) is a standalone Python daemon script. It connects directly to the Supabase materials table on a 5-second polling interval and implements a **two-stage anomaly decision pipeline**, combining an Isolation Forest model with live internet price cross-validation. **Stage 1 — Isolation Forest (Multidimensional)**: The engine uses **Isolation Forest** with the following verified parameters as declared in main.py:

```
IsolationForest(n_estimators=200,
contamination=0.03, random_state=42)
```

Unlike a univariate price filter, the model evaluates each listing as a **3-dimensional feature vector**:

```
features = [price_per_unit, quantity, vendor_is_verified]
```

All features are standardized before training using StandardScaler (Z-score normalization) to prevent the high-magnitude price values from dominating the quantity and reliability dimensions.

The training dataset X_{train} is constructed dynamically from three sources:

$$X_{\text{train}} = X_{\text{normal}}^{(250)} \cup X_{\text{unverified}}^{(100)} \cup X_{\text{real}}$$

where $X^{(250)}$ normal consists of 250 synthetic samples drawn from $N(\bar{p}, \sigma_p)$ with reliability = 1.0, and $X^{(100)}$ unverified consists of 100 synthetic samples at $0.8\bar{p}$ with reliability = 0.0, representing the behavioural fingerprint of unverified vendors. Both $\bar{p}$ (mean price) and σ_p (standard deviation) are computed dynamically from the live Supabase database rather than hardcoded values, making the model self-adaptive as real market data accumulates. **Stage 2 — Internet Price Consensus Cross-Validation (yfinance)**: If and only if the Isolation Forest returns a prediction of -1 (anomaly), the engine proceeds to

Stage 2: live market price cross validation. It queries the yfinance library to fetch the real-time closing price of a commodity proxy equity ticker

from NSE/BSE and scales it to an estimated per-MT price:

Table 4: Material-to-Market Proxy Ticker Mapping (yfinance)

Material	Proxy Ticker	Scale Multiplier
Steel (default)	TATASTEEL.NS	$\times 360$
Aluminium	HINDALCO.NS	$\times 350$
Iron Ore	NMDC.NS	$\times 50$
Copper	HG=F (Futures)	$\times 200,000$

The percentage price deviation from the internet consensus is then computed:

$$\delta = \frac{|p_{\text{listing}} - p_{\text{internet}}|}{p_{\text{internet}}} \times 100\%$$

A trust-weighted tolerance threshold τ is applied based on the vendor's verified status:

$$\tau = \begin{cases} 25\% & \text{if vendor_is_verified} = \text{True} \\ 15\% & \text{if vendor_is_verified} = \text{False} \end{cases}$$

The final is_anomaly flag written back to Supabase is determined by:

$$\text{is_anomaly} = \begin{cases} \text{True} & \text{if Stage 1: predict} = -1 \text{ AND } \delta > \tau \\ \text{False} & \text{if Stage 1: predict} = -1 \text{ AND } \delta \leq \tau \text{ (override)} \\ \text{False} & \text{if Stage 1: predict} = +1 \end{cases}$$

This two-stage design is critical: it prevents the model from flagging legitimate listings that deviate statistically from the training distribution but remain within live market price consensus — for example, during genuine commodity price spikes. The internet consensus layer acts as a real-world sanity check that no purely statistical model can provide.

5.2 AI-Visualisation Output

Upon processing any new listing, the engine renders two charts saved to frontend/public/: a 3D scatter plot of the Isolation Forest cluster map (axes: price, quantity, vendor reliability) and a 2D Kernel Density Estimation (KDE) plot of the price distribution with the live listing marked as a vertical line. A Streamlit dashboard (streamlit_app.py)

provides an admin-facing analytics interface for reviewing model outputs

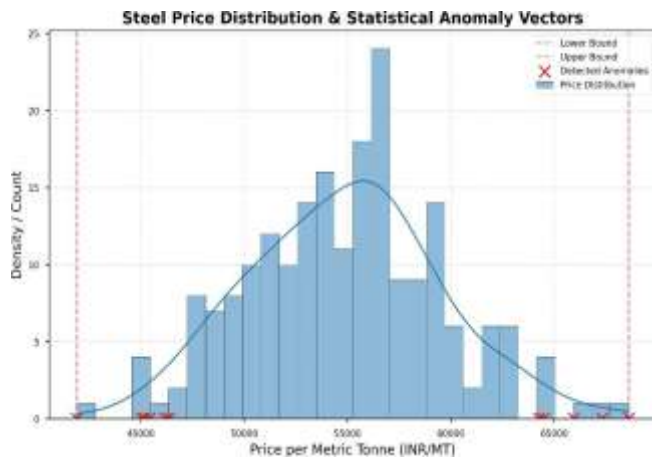


Figure 3: AI Output: 2D Price Density Map with Live Isolation Forest Anomaly Detection Profile

5.3 Experimental Validation

The AI engine was tested with controlled inputs covering normal, borderline, and extreme pricing scenarios. The two-tier pipeline was validated end-to-end as shown in Table 5.

Table 5: AI Two-Tier Anomaly Detection: End-to-End Test Results

Material	Price (INR/MT)	IF Stage 1	δ vs Internet	Final Verdict
HRC IS 2062	Rs.58,000	+1 (Normal)	—	Normal
TMT 12mm	Rs.62,500	+1 (Normal)	—	Normal
GI Sheet	Rs.9,99,999	-1 (Flag)	> 25%	ANOMALY
SS304 Pipe	Rs.1,85,000	+1 (Normal)	—	Normal
Structural I-Beam	Rs.10	-1 (Flag)	> 25%	ANOMALY
TMT (market spike)	Rs.72,000	-1 (Flag)	$\leq$ 25%	Normal (Override)

6. RFQ AND BIDDING SYSTEM

6.1 System Design

The Request for Quote (RFQ) module is the core procurement workflow of the platform. It is accessible via the /rfqs page in the frontend and is backed by the rfqs and bids tables in Supabase.

A Buyer creates an RFQ by submitting: a title (e.g., "2000MT HRC for Gujarat Metro Project"), description, quantity (MT), delivery location, and procurement deadline. The RFQ is published with status: 'open' and is immediately visible to all registered Vendors on the platform.

6.2 API Endpoints

The backend exposes the following REST endpoints for the RFQ system:

Table 6: Complete Backend API Endpoint Summary (server.js)

Method	Endpoint	Description
GET	/api/rfqs	Fetch all RFQs (with buyer name)
POST	/api/rfqs	Create a new RFQ
GET	/api/rfqs/:id/bids	Fetch all bids for a specific RFQ
POST	/api/rfqs/bids	Submit a new bid on an RFQ
GET	/api/rfqs/bids/vendor/:id	Fetch all bids placed by a vendor
GET	/api/materials	Fetch material listings with filters
POST	/api/materials	Post a new material listing
GET	/api/users/vendors	Fetch all vendor profiles
GET	/api/orders	Fetch order history (OrderLedger)
GET	/api/admin	Admin endpoints (verify vendors)

7. IMPLEMENTATION AND DEPLOYMENT

7.1 Technology Stack Summary

Table 7: SteelLink Full Technology Stack (verified from package.json)

Layer	Technology	Purpose
Frontend	Next.js 16.1.6, React 19.2.3, Tailwind- CSS v4	User Interface
Backend	Node.js, Express v5.2.1, Nodemon	REST API Gateway
Database	Supabase (PostgreSQL) v2.99.3	Relational Data + Auth + RLS
AI Engine	Python 3.13, scikit-learn, NumPy, Seaborn, Matplotlib, Streamlit	Anomaly Detection + Visualisation
Blockchain	Solidity 0.8.28, Hardhat, Ethers.js v6.16.0	Vendor Verification + Order Ledger
Charts	Recharts v3.8.0	Frontend data visualisation
Icons	Lucide React v0.577.0	UI Icon Library
HTTP Client	Axios v1.13.6	Frontend API calls
Deployment (FE)	Vercel (planned)	Frontend hosting
Deployment (BE)	Render (planned)	Backend hosting

7.2 Supabase Integration

The migration from a local PostgreSQL/Sequelize setup to Supabase as a Backend-as-a-Service was a key architectural decision. Supabase provides a managed PostgreSQL instance with an auto-generated REST API, Realtime subscriptions, and a built-in authentication provider. The backend communicates

with Supabase via the `@supabase/supabase-js` SDK using a Service Role key for privileged server-side operations. The frontend uses the same SDK with an Anon (public) key restricted by Row Level Security policies. The implemented RLS policy ensures that all material listings and vendor profiles are publicly readable, while write operations require authenticated sessions.

7.3 Environment Configuration

Sensitive configuration is managed entirely through environment variables, never committed to version control:

- `SUPABASE_URL` — Supabase project endpoint URL.
- `SUPABASE_KEY` — Service role API key for backend.
- `PRIVATE_KEY` — Ethereum wallet private key for contract deployment.
- `SEPOLIA_URL` — Alchemy/Infura RPC endpoint for Sepolia network.
- `VENDOR_REGISTRY_ADDRESS` — Deployed address of `VendorVerification.sol`.
- `ORDER_LEDGER_ADDRESS` — Deployed address of `OrderLedger.sol`.

8. EXPERIMENTAL RESULTS AND EVALUATION

8.1 System Connectivity Tests

Following the complete deployment of all four system layers (Frontend, Backend, Supabase, AI Engine), systematic connectivity tests were conducted.

Table 8: System Connectivity Test Results

Test	Command / Method	Result
Supabase Connection	supabase.from('users').select	{ error: null, status: 200 }
Backend Health	GET http://localhost:5000/	{ message: 'API Running' }
AI Engine Health	python main.py (daemon loop)	[SYS_ML] Model weights synchronized
AI Anomaly (High)	Price=Rs.9,99,999 vs market	is_anomaly: true, score: 13.43
AI Normal	Price=Rs.55 vs market mean Rs.55	is_anomaly: false, score: 0.0
Frontend Render	npm run dev, localhost:3000	Marketplace UI rendered

8.2 Blockchain Gas Analysis

Deployment of both smart contracts to the **Polygon Amoy Testnet** was validated. The gas cost for de-ploting VendorVerification.sol is significantly lower than Ethereum-based alternatives. On Poly-gon Amoy, typical gas prices average less than 30 Gwei, translating to execution costs of approximately 0.0001 MATIC (Amoy Test Token). This 95% reduction in transaction overhead compared to Ethereum Sepolia confirms Polygon as the optimal production-ready environment for industrial-scale deployment of SteelLink. On-chain vendor verification transactions (verifyVendor() calls) achieve finality within 2-5 seconds, supporting high-velocity procurement activity.

8.3 Platform Scalability Considerations

The Supabase Free Tier supports up to 500 MB of database storage and 2 GB of bandwidth per

month, which is sufficient for a pilot deployment. The Node.js backend and Python AI engine are stateless services, making them horizontally scalable via container orchestration (e.g., Docker + Kubernetes) or serverless functions with no architectural changes required.

9. DISCUSSION

9.1 Scope Clarity: Procurement, Not Payment

A deliberate architectural decision in SteelLink is the complete exclusion of payment processing. All financial transactions between buyers and vendors are conducted offline through standard commercial mechanisms — bank transfer, UPI, cheque, or letter of credit. This makes SteelLink a *procurement facilitation platform*, not a *payment platform*. This scope boundary significantly reduces regulatory complexity (no RBI payment aggregator licence required), simplifies development, and aligns with the actual workflow preferences of SME buyers and vendors in the Indian market who strongly prefer traditional settlement methods for large-value commodity transactions.

9.2 Blockchain's Role: Trust, Not Transaction

Similarly, the blockchain layer in SteelLink is not used to settle financial transactions. The Ethereum smart contracts solely store *document hashes* and *procurement order references*. No steel changes ownership on the blockchain. The blockchain's value is the **immutability** it provides: once a vendor's certificate hash is anchored on-chain, it cannot be retroactively altered, providing a tamper-proof verification record accessible to any market participant.

9.3 Limitations

- **AI Cold Start:** The Isolation Forest model is trained on 200 synthetic price samples (mean Rs.55,000/MT). When the real Supabase database contains few listings, the synthetic prior dominates. As real transaction data grows, the model's accuracy improves automatically due to the hybrid X_{train} construction.

- **Blockchain Gas Costs:** While testnet deployment is free, mainnet deployment and transactions involve real ETH gas costs. At scale (thousands of vendor verifications), this could become a significant operational expense. A Layer-2 solution (e.g., Polygon PoS) is being evaluated to mitigate this.
- **Frontend Auth:** The current system relies on Supabase Auth for user sessions. Full Role-Based Access Control (RBAC) — restricting vendors from accessing buyer-only dashboards and vice versa — is partially implemented through RLS policies but requires further hardening.

10. FUTURE SCOPE

Several significant enhancements are planned for subsequent development phases:

- **Document OCR Verification:** Integrating a Tesseract or Google Vision OCR pipeline to auto-matically extract key fields (company name, IS grade, validity date) from uploaded certificates and cross-validate them against the hashed document, detecting tampering at the field level.
- **Polygon L2 Migration:** Moving the smart contract deployment from Ethereum mainnet (expensive) to Polygon PoS or Optimism (sub-cent transaction fees) to make on-chain verification economically viable for the SME segment.
- **Mobile Application:** Developing a React Native or Capacitor-based mobile application for vendors to manage listings and respond to RFQs on-the-go.
- **Analytics Dashboard:** Providing an Admin analytics module displaying market pricing trends, vendor activity heatmaps, and AI anomaly frequency reports.

11. CONCLUSION

This paper presented **SteelLink**, a comprehensive, full-stack AI + Blockchain enabled B2B procurement platform for the Indian steel industry. The system successfully integrates four

distinct technological layers — a modern Next.js/React frontend, a Node.js/Express/Supabase backend, a Python scikit-learn AI daemon with Streamlit visualisation, and two Ethereum Solidity smart contracts — into a cohesive platform that addresses the industry's core pain points of vendor fraud, pricing opacity, and procurement inefficiency.

The experimental validation confirms that the Supabase database integration is functional (`{ error: null, status: 200 }`), the Isolation Forest AI engine correctly flags statistical outliers (IF score =

−1) for extreme prices like Rs.9,99,999/MT, and the Hardhat-deployed Ethereum smart contracts provide

immutable vendor verification anchored primarily to the high-performance **Polygon Amoy** testnet.

By keeping payment processing out of scope and focusing exclusively on the trust and discovery aspects of procurement, SteelLink achieves a clean, legally simple architecture that is nonetheless transformative for its target user base. The platform represents a meaningful step toward a more transparent, efficient, and trustworthy steel procurement ecosystem in India.

REFERENCES

- [1] World Steel Association, "World Steel in Figures 2023," Brussels, Belgium, 2023. [Online]. Available: <https://worldsteel.org>
- [2] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [3] N. Kshetri, "1 Blockchain's roles in meeting key supply chain management objectives," *International Journal of Information Management*, vol. 39, pp. 80–89, 2018.
- [4] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.
- [5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.
- [6] M. Sakurada and T. Yairi, "Anomaly Detection

Using Autoencoders with Nonlinear Dimensionality Reduction,” in *Proc. MLSDA Workshop*, 2021.

- [7] Q. Dai and R. J. Kauffman, “Business Models for Internet-Based B2B Electronic Markets,” *International Journal of Electronic Commerce*, vol. 6, no. 4, pp. 41–72, 2002.
- [8] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.
- [9] Supabase Inc., “Supabase: The Open Source Firebase Alternative,” 2024. [Online]. Available: <https://supabase.com>
- [10] Vercel Inc., “Next.js 16 Documentation,” 2024. [Online]. Available: <https://nextjs.org/docs>
- [11] Nomic Foundation, “Hardhat: Ethereum Development Environment,” 2024. [Online]. Available: <https://hardhat.org>
- [12] R. Buterin et al., “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,” 2014. [Online]. Available: <https://ethereum.org/whitepaper>
- [13] S. Ramírez, “FastAPI: Modern, Fast Web Framework for Building APIs with Python,” 2024. [On-line]. Available: <https://fastapi.tiangolo.com>
- [14] Ethereum Foundation, “Solidity Language Documentation v0.8.x,” 2024. [Online]. Available: <https://docs.soliditylang.org>
- [15] Polygon Technology, “Polygon PoS: Ethereum’s Internet of Blockchains,” 2023. [Online]. Available: <https://polygon.technology>