Title
Serve at Ease: An AI-Driven Federated Learning and Trust Scoring Framework for Service Platforms

Authors
Tarun M , Vinayaka V M, Swayam Sidnale, Vivek Gunari

Affiliation
Department of Computer Science and Engineering, Dayananda Sagar College of Engineering, Bengaluru, Karnataka, India Institution, City, Country

Email
author1@example.com, author2@example.com

Abstract:
This paper presents Serve at Ease, a privacy-first framework that integrates federated learning with a transparent trust-scoring pipeline for service platforms to mitigate data exposure, bias, and fraud while maintaining scalability. Unlike centralized analytics that aggregate raw user and vendor data, the system coordinates decentralized model training on edge nodes and exchanges only anonymized model updates for aggregation, aligning with modern FL practices. The literature review synthesizes advances in federated optimization, trust and reputation modeling, and bias mitigation in marketplace platforms, then operationalizes these concepts for real-time fraud alerts and role-aware analytics. A partially implemented prototype demonstrates role-based interfaces (customer, vendor, admin), secure authentication, baseline schemas for trust and model states, and an orchestrator for update intake and global model distribution. The approach is designed to enhance user confidence, reduce manipulation, and align with deployment constraints, while enabling future integration of blockchain-backed update attestation and large-scale evaluation.

Keywords:
Federated Learning; Trust Scoring; Fraud Detection; Privacy; Service Platforms; Edge Computing; Reputation Systems.

1. Introduction
Service marketplaces depend on reliable interactions among customers, vendors, and platform operators, yet centralized data pipelines amplify risks such as privacy leakage, adversarial manipulation, and latency in fraud response. Conventional machine learning pipelines typically require central collection of behavior logs and transactions, which in turn increases compliance burden and single-point failures. Federated learning (FL) addresses these concerns by training models at the data origin and transmitting learned parameters for secure aggregation, thus limiting raw data movement while preserving utility. Serve at Ease applies this principle to trust and fraud analytics in service platforms, providing a streamed, privacy-preserving alternative to traditional reputation engines. The contributions are threefold: a literature-anchored system design for FL-based trust scoring in marketplaces; an orchestrated backend enabling secure update flow and global model distribution; and an implementation-in-progress showcasing role-based dashboards, authentication, and baseline data schemas for trust and models.

2. Literature review
2.1 Federated learning foundations
Federated learning introduced decentralized optimization where clients train locally and share model updates to an aggregator, reducing raw data exposure. Subsequent work generalized FL to domains

with strict privacy and compliance requirements, including healthcare, finance, and IoT, emphasizing communication efficiency, client heterogeneity, and robustness. These advances motivate applying FL to marketplaces in which user and vendor data sensitivity is high and participation is dynamic.

2.2 Trust, reputation, and fraud analytics
Trust computation has evolved from probabilistic and evidence-based formulations to deep behavioral models that infer reliability from interactions, temporal patterns, and anomalies. Reputation systems in consumer platforms face manipulation, selection bias, and echo effects, and centralized curation can introduce opaque decisions. Integrating trust models with streaming anomaly detection is beneficial for early fraud signals and adaptive scoring.

2.3 FL combined with trust scoring
Emerging studies propose merging decentralized learning with trust/reputation signals to reduce data exposure and limit systemic bias from centralized curation. Aggregation-time defenses, differential privacy, and secure aggregation help resist gradient leakage and model inversion, while robust aggregation can mitigate poisoned updates from malicious clients. For service platforms, this integration aims to provide privacy-aware evaluations with resilience to gaming and model drift.

2.4 Gaps and implications for marketplaces
Across the literature, observed gaps include limited end-to-end prototypes with real user roles, sparse reporting on admin-facing fraud workflows, and insufficient life-cycle governance of models and reputational feedback loops. Serve at Ease addresses these gaps by combining federated training orchestration, trust score computation, and an admin dashboard for monitoring anomalies and model behaviour, providing a practical bridge from theory to deployment.

3. Proposed system architecture

Overview
Serve at Ease coordinates local model training on user and vendor devices, transmits only encrypted gradients/weights, and aggregates updates to refine a global model that supports trust scoring and fraud alerts. The platform returns updated parameters and trust metrics to clients via secure APIs, enabling privacy-preserving, iterative improvements.

Key modules

- Data locality: Devices retain logs such as bookings, ratings, reviews, and behavioural signals to ensure privacy and regulatory alignment.
- Local model training: Lightweight models learn on-device from behavioural data to identify trustworthy and anomalous patterns.
- Update sharing: Clients send encrypted model updates rather than raw records, reducing leakage risk.
- Global aggregation: The backend orchestrator aggregates client updates and produces a refreshed global model.
- Trust scoring: The platform computes trust scores informed by consistency, completion quality, feedback integrity, and anomaly signals.

Technologies

- Frontend: React + Vite

- Backend: Node.js + Express
- Database: MongoDB (User, TrustScore, GlobalModel, LocalUpdate)
- Federated AI: TensorFlow Federated / PySyft (simulations during development)
- Security: JWT authentication, encrypted transport for updates

System flow
User/Vendor Device → Local Model (on private data) → Encrypted Updates → Federated Orchestrator (Aggregation) → Updated Global Model → Trust Score Update at Client/Server

4. Methodology

4.1 Federated orchestrator (backend API)
The orchestrator mediates update intake and distribution of the current global model snapshot. Clients submit updates through a secure endpoint and retrieve the latest global parameters when eligible. Aggregation strategies can be configured (e.g., weighted averaging, robust aggregation) and instrumented for audit and rollback. Example endpoints:

- POST /federated/update — submit encrypted gradients/weights
- GET /federated/global-model — fetch the latest global model state

4.2 Trust scoring engine
The engine computes user/vendor trust scores using multi-factor signals: transaction completion rates, dispute/cancellation ratios, timeliness/service quality, feedback credibility, and anomaly indicators. Scores are recalibrated as the global model improves. Example endpoint:

- GET /trust-score/:id — retrieve current trust score and summary factors

4.3 Authentication and role management
A JWT-based access layer enforces roles for customers, vendors, and admins. Core endpoints:

- POST /auth/register — create account with role binding
- POST /auth/login — obtain JWT token for protected APIs

4.4 Admin control panel
The admin dashboard surfaces:

- Live system statistics and user/vendor activity
- Real-time fraud alerts and anomaly feeds
- Global model performance summaries and update volumes
- Aggregated trust trends and segments

4.5 Model integration and analytics
Python-backed models (e.g., XGBoost, TensorFlow) support:

- Fraud likelihood scoring
- Demand forecasting for services
- Vendor performance evaluation and drift checks
  Model updates are logged with version metadata to support rollbacks, audits, and A/B tests.

5. Implementation progress and results

Frontend (React + Vite)

- Customer dashboard: service discovery, bookings, order management, profile with trust placeholder.
- Vendor dashboard: service creation/management, profile customization, trust visualization placeholders.
- Admin dashboard: monitoring layout with analytics placeholders for fraud alerts and model metrics.
- UI stack: React Router for role navigation; Tailwind CSS/Material UI for responsive design; mock APIs for interaction testing.

Backend (Node.js + Express + MongoDB)

- REST APIs scaffolded for authentication, trust score access, and federated endpoints.
- Schemas: User, TrustScore, LocalUpdate, GlobalModel with version/time metadata.
- Environment prepared for secure transport and token-based access control.

Federated learning simulation (in progress)

- TFF/PySyft setup for local update simulation and aggregation loop testing.
- Placeholder pipelines for model distribution and client eligibility checks.

Testing and deployment plan

- Integration testing with Postman and scripted test clients.
- MongoDB Atlas for managed persistence; planned deployment on Vercel (frontend) and Render/AWS EC2 (backend).
- Observability to track update rates, aggregation cycles, and anomaly yields.

Observed outcomes so far

- Role-based UI flows validated with mock data.
- Endpoints and schemas enable stitching of trust and FL loops.
- Admin-facing surfaces scoped for fraud alerts and model health.

6. Discussion

The partial implementation indicates feasibility of layering FL-based trust scoring atop existing marketplace mechanics while retaining privacy and modularity. Practical challenges include client heterogeneity, intermittent participation, and communication overhead. Robust aggregation, rate limiting, and privacy-preserving defenses are critical to resist poisoned updates and leakage while maintaining model utility. Governance—the policies that tie scores to platform actions—must be transparent to minimize unfair penalties or bias amplification.

7. Conclusion and future work

Serve at Ease operationalizes a privacy-preserving trust and fraud analytics stack by combining

federated learning with interpretable scoring and admin analytics. The architecture reduces centralized exposure, supports iterative improvement via aggregated updates, and provides actionable insights for platform integrity. Future work includes evaluating robust aggregation under targeted poisoning and sybil scenarios, integrating secure aggregation and calibrated differential privacy noise for update protection, and exploring blockchain-backed update attestations to strengthen auditability. Large-scale A/B tests and user studies will quantify trust impacts, false-positive/negative trade-offs, and long-term fairness effects in production deployments.

8. References:

H. B. McMahan, E. Moore, D. Ramage, et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," AISTATS, 2017.

T. Yang, G. Andrew, H. Eichner, et al., "Applied Federated Learning: Improving Google Keyboard Query Suggestions," arXiv:1812.02903, 2018.

P. Kairouz, H. B. McMahan, B. Avent, et al., "Advances and Open Problems in Federated Learning," Foundations and Trends in Machine Learning, 2021.

A. Jøsang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," Decision Support Systems, 2007.

Y. Zhang, X. Chen, and J. Chen, "Trustworthy AI: From Principles to Practices," ACM Computing Surveys, 2020.

L. Chen, S. Mislove, and A. Wilson, "An Empirical Analysis of Algorithmic Bias in Peer-to-Peer Platforms," WWW, 2022.

J. Liu, Z. Wu, and M. Whinston, "Manipulation and Bias in Platform Reputation Systems," Management Science, 2023.

K. Bonawitz, H. Eichner, W. Grieskamp, et al., "Towards Federated Learning at Scale: System Design," MLSys, 2019.

P. Blanchard, E. Mhamdi, R. Guerraoui, et al., "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," NeurIPS, 2017.

N. Papernot, M. Abadi, U. Erlingsson, et al., "Semi-Supervised Knowledge Transfer for Deep Learning from Private Training Data," ICLR, 2017.