© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1717808

AI-Powered Local Service Booking and Verification Web Application Using Face Authentication and Smart Recommendation System

M. SIVANESWARA PERUMAL[1], M. ABDUL RAHMAN [2], P. TAMILARASAN [3], DR. R. DHAMODHARAN[4]

[1, 2, 3]*UG Student Dept. Of Computer Science Kalasalingam Academy of Research and Education, Tamil Nadu, India*

[4]*Assistant Professor Dept. Of Computer Science and It Kalasalingam Academy of Research and Education, Tamil Nadu, India*

Abstract- ***The increasing demand for trusted local service providers — including plumbers, electricians, carpenters, and housekeeping staff — has exposed critical gaps in current booking platforms: fake service registrations, absence of identity verification, poor trust mechanisms, and delayed emergency service response. This paper presents the design and implementation of an AI-powered local service booking and verification web application that addresses these challenges through a multi-layered approach. The proposed system integrates face authentication using TensorFlow and Face API with government-issued ID verification to eliminate fraudulent registrations. A smart recommendation engine, built using collaborative and content-based filtering, suggests service providers to users based on ratings, location, and past booking behaviour. The platform supports real-time booking and scheduling, secure payment processing via Razorpay, a dynamic rating and review system, and emergency service request handling with priority routing. The backend is developed using Node.js and Express.js, the frontend with React.js, and MongoDB serves as the primary database. Google Apps Script is used for automated email notifications, and Firebase handles cloud storage for identity documents. Preliminary results demonstrate a face verification accuracy of 94.7%, a booking response time below 1.8 seconds, and a fraud prevention rate exceeding 91%. The platform provides a reliable, scalable, and low-cost solution for modernising local service ecosystems.***

Keywords — Local Services, AI Verification, Face Authentication, Smart Recommendation, Service Booking, Emergency Services, Fraud Detection, Web Application, Google Apps Script, Machine Learning.

I. INTRODUCTION

The rapid growth of urban populations across India and other developing nations has intensified the need for accessible, trustworthy local service platforms. Millions of households require daily services such as electrical repairs, plumbing, home cleaning, painting, and carpentry. Yet, identifying a qualified, genuine, and reliable service professional remains a persistent challenge. According to the Ministry of Skill Development and Entrepreneurship (India, 2022), over 400 million workers are employed in the informal service sector, and more than 65% of consumers report difficulty in verifying the authenticity and qualifications of local service workers.

Traditional methods of finding service providers — word-of-mouth referrals, printed directories, or unverified online classifieds — are inherently unreliable. Fraud cases involving fake workers who misrepresent their skills, steal from customers, or cause property damage have been widely reported. A 2021 consumer survey by LocalCircles found that 42% of Indian urban households had experienced at least one incident involving an unverified or fraudulent home service provider. These figures highlight a genuine public safety concern that existing mainstream platforms have not adequately addressed.

Existing digital platforms such as Urban Company, Housejoy, and TaskRabbit provide an online marketplace for local services, but they have notable

limitations. Most platforms rely on self-reported credentials without independent identity or skill verification. Their recommendation algorithms do not account for real-time location proximity or user preference history effectively. Furthermore, none of these platforms offers emergency service routing, which is critical when a household faces a gas leak, power failure, or plumbing emergency at odd hours.

This paper presents an AI-powered local service booking and verification web application that addresses these shortcomings. The system introduces three core innovations: (1) biometric face authentication combined with government-issued ID verification to prevent fraudulent registrations; (2) a machine-learning-based smart recommendation engine that suggests suitable providers based on user history, ratings, and geographic proximity; and (3) an emergency request module that routes urgent service needs to the nearest verified available provider. The platform also integrates secure payment processing, a transparent review system, and automated email notifications.

The rest of this paper is structured as follows. Section II reviews related literature. Section III describes the proposed system. Section IV presents the architecture. Section V explains the methodology. Section VI covers module testing. Section VII discusses results and analysis. Sections VIII to X cover advantages, future scope, and conclusions, followed by references.

II. RELATED WORK

Research in digital service booking, identity verification, recommendation systems, and fraud detection provides a strong foundation for the proposed system. This section reviews key literature in each area.

A. Online Service Booking Platforms

Yoo et al. [1] analysed the design principles of two-sided service marketplaces and concluded that trust signals — verified credentials, public ratings, and transparent pricing — are the most significant determinants of consumer platform adoption. Malhotra and Malhotra [2] studied the growth trajectory of Urban Company and found that while digital booking improved accessibility, the absence of formal background checks created recurring safety complaints. Their work emphasised the need for automated verification systems integrated into the registration workflow.

B. Face Recognition and Biometric Verification

Taigman et al. [3] introduced DeepFace, a deep learning model achieving near-human face verification accuracy on the LFW benchmark. Subsequent work by Parkhi et al. [4] on VGGFace demonstrated that large-scale training datasets combined with convolutional neural networks produce highly robust face embeddings suitable for identity verification. Jain and Kumar [5] provided a comprehensive survey of face recognition in identity verification systems and highlighted the importance of liveness detection in preventing spoofing attacks. These techniques underpin the face verification module in the proposed system.

C. Recommendation Systems

Collaborative filtering and content-based filtering are the two dominant paradigms in recommendation system literature [6]. Koren et al. [7] established matrix factorisation as a powerful collaborative filtering technique. For service recommendation, Zheng et al. [8] demonstrated that hybrid approaches combining user-item interaction history with contextual signals such as location and time-of-day substantially improve the relevance of service suggestions compared to single-method approaches. This hybrid strategy was adopted in the proposed recommendation engine.

D. Fraud Detection in Online Platforms

Zhang et al. [9] proposed a graph-based fraud detection model for e-commerce platforms that identifies suspicious registration clusters by analysing behavioural and network patterns, achieving 93% precision in flagging fraudulent accounts. Liu et al. [10] applied anomaly detection algorithms to online marketplace data and demonstrated that combining document verification with behavioural analysis reduced fake provider registrations by 87%. These findings directly motivate the multi-layer verification approach in the proposed system.

E. Emergency Response Systems

Hasan et al. [11] reviewed digital emergency response frameworks and found that location-aware priority routing reduces average emergency service response time by 34% compared to manual dispatch. Chen and Wang [12] demonstrated that real-time geospatial matching between emergency requests and available responders, combined with automated notifications, produces measurable reductions in service delay times. These insights inform the design of the emergency request module.

F. Research Gaps

No existing system integrates AI-based face authentication, government ID cross-verification, smart recommendation with location awareness, emergency routing, and secure payment in a single platform targeting the informal local services sector. The proposed system addresses this gap comprehensively.

III. PROPOSED SYSTEM

The proposed system is a full-stack web application providing a trusted digital marketplace for local service providers and consumers. The platform serves three primary user roles: customers who book services, service providers who register and accept bookings, and administrators who manage verifications and monitor platform activity.

A. User and Provider Registration

Customers register using name, email, phone number, and address. Service providers complete an extended registration process that includes uploading a government-issued photo ID (Aadhaar card, driving licence, or passport), providing skill credentials, and completing a face verification step. The system captures a live selfie via webcam and compares the facial embedding against the ID photograph using Face API and TensorFlow. Registration is only approved when the face match confidence exceeds a threshold of 0.75 similarity and the ID document is confirmed as valid and non-duplicate.

B. Smart Recommendation Engine

The recommendation engine employs a hybrid filtering approach. Content-based filtering profiles each provider based on service category, location, average rating, experience, and response time. Collaborative filtering analyses historical booking patterns to identify providers preferred by users with similar profiles. The engine computes a composite relevance score for each provider relative to the requesting customer's location, past preferences, and time of day. Results are ranked and presented with human-readable explanations such as 'Highly rated in your area' or 'Previously booked by users like you'.

C. Booking, Scheduling, and Payments

Customers can view provider profiles, check real-time availability, and book appointments for specific time slots. Bookings trigger automated confirmation emails to both parties via Google Apps Script and Gmail. Payment is processed through Razorpay, supporting UPI, debit or credit cards, and net banking. Receipts are auto-generated and stored in MongoDB. Providers can accept, reschedule, or decline bookings through their personal dashboard.

D. Emergency Request Module

The emergency module accepts urgent service requests specifying the nature of the problem and the customer's current location. The system immediately identifies all verified providers in the matching service category within a configurable radius and sends simultaneous push notifications and email alerts. The first provider to accept is assigned, and the customer is notified in real time with provider details and an estimated arrival time. Emergency requests are prioritised over standard bookings.

E. Rating, Review, and Admin Panel

After service completion, customers submit a rating (1 to 5 stars) and written review. Ratings are aggregated and publicly displayed on provider profiles. Providers with average ratings below 3.0 are flagged for admin review. The admin panel shows total bookings, verified provider counts, pending verifications, flagged accounts, and system-wide metrics. Administrators can approve or reject provider registrations, suspend accounts, and manage disputes.

IV. ARCHITECTURE

The system follows a modular three-tier architecture comprising a React.js frontend, a Node.js and Express.js backend, and a MongoDB database, with integrated AI modules and third-party services communicating via REST APIs.

A. Frontend Layer

The frontend is developed using React.js with functional components and React Hooks for state management. The UI is styled with CSS modules and is responsive across desktop, tablet, and mobile devices. Key pages include the home and search page, provider profile view, booking and payment flow, customer and provider dashboards, emergency request form, and admin control panel. Axios handles API communication with the backend. The face capture module uses the browser MediaDevices API to access the webcam and sends frames to the verification endpoint.

B. Backend Layer

The backend uses Node.js with Express.js and exposes a RESTful API. Key services include JWT-based authentication, provider verification (face comparison and ID validation), booking management, recommendation scoring, payment processing via Razorpay API, and emergency request routing. All endpoints are protected by middleware for authentication and role-based access control. Sensitive identity documents are stored via Firebase Cloud Storage.

C. Database Layer

MongoDB is used as the primary database with collections for users, providers, bookings, reviews, emergency requests, and notifications. Indexes are maintained on frequently queried fields such as provider location coordinates using 2dsphere indexing for geospatial queries, service category, and booking status. This ensures fast query response times even as data volume grows.

D. AI and Verification Modules

The face authentication module uses face-api.js, a TensorFlow.js-based library providing pre-trained models for face detection and 128-dimensional face descriptor extraction. The system computes facial descriptors for the ID photograph and the live selfie and calculates the Euclidean distance between them. A distance below 0.6 is treated as a match. The recommendation engine is implemented as a Node.js service that computes relevance scores using pre-aggregated provider features stored in MongoDB.

E. Notification and Payment Services

Google Apps Script is deployed as a web app endpoint that the backend calls to send transactional and emergency emails via Gmail SMTP. Razorpay handles all payment transactions with sandbox and production API environments. Payment events such as success, failure, and refund are captured via Razorpay webhooks and stored in MongoDB for audit and reconciliation purposes.

V. METHODOLOGY

The system was developed following an iterative Agile methodology across requirement analysis, system design, module development, integration, testing, and deployment phases.

A. Requirement Analysis and Planning

A structured requirements study was conducted involving a survey of 120 urban households and interviews with 30 freelance service workers in Chennai, Tamil Nadu. The survey identified the top three pain points as inability to verify provider identity (78% of respondents), lack of emergency availability (61%), and absence of reliable ratings (55%). These findings shaped the feature priorities of the system.

B. System and Database Design

Entity-relationship modelling was performed to design the MongoDB schema. Use case diagrams were created for all three user roles. The recommendation engine scoring formula was derived by reviewing hybrid filtering literature. The face verification threshold was calibrated by testing face-api.js on a dataset of 500 image pairs (250 matching, 250 non-matching), selecting the threshold that maximised the F1 score.

C. Frontend and Backend Development

Frontend components were developed module by module using React.js. API contracts between frontend and backend were defined using Postman before implementation. The Express.js backend was developed with a controller-service-repository pattern to ensure separation of concerns. Middleware for JWT validation, file upload handling with Multer, and rate limiting with express-rate-limit was integrated at the router level. Face-api.js models were loaded from CDN on the provider registration page.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1717808

D. AI Model Integration and Testing

The face verification pipeline was tested under three lighting conditions: clear natural light, indirect indoor light, and low light. Accuracy was measured at 97.2%, 94.7%, and 88.3% respectively. A lighting requirement notice is shown on the registration page to guide users. The recommendation engine was evaluated on a held-out test set of 200 historical bookings. The hybrid model achieved a precision at k=5 of 0.82, compared to 0.67 for the content-only baseline.

E. Deployment

The frontend was deployed on Vercel for continuous delivery. The Node.js backend was deployed on Railway.app. MongoDB Atlas served as the cloud database. Firebase was configured for identity document storage. Google Apps Script was published as a restricted web app endpoint. Environment variables were managed using Railway's secret management to avoid credential exposure.

VI. MODULE DESCRIPTION AND TESTING

A. Module Descriptions

Table I. Module Descriptions

No.	Module Name	Description
1	Registration Module	Handles customer and provider sign-up. Provider flow includes ID upload, face capture, and two-factor verification before account activation.
2	Verification Module	Performs face-to-ID matching using face-api.js. Rejects duplicate IDs using hash comparison. Flags low-confidence matches for admin review.
3	Booking Module	Manages service appointments, time-slot availability, booking confirmation, and status tracking: pending, confirmed, completed, or cancelled.
4	Recommendation Module	Computes hybrid relevance scores combining content-based provider features and collaborative user-booking patterns to rank providers for each search.
5	Emergency Module	Accepts priority service requests, locates the nearest available verified providers, sends simultaneous alerts, and assigns the first accepting responder.
6	Payment Module	Integrates Razorpay for UPI, card, and net banking payments. Handles payment confirmation and failure recovery via webhook events.
7	Admin Module	Provides a management dashboard for verifying providers, reviewing flagged accounts, resolving disputes, and viewing system-wide analytics.

B. Unit Testing

Each module was subjected to independent unit testing before integration. Table II summarises the key test cases and outcomes.

Table II. Unit Testing Results

Module	Test Case	Expected Result	Status
Registration	Valid user submits registration form	Account created, confirmation email sent	PASS
Verification	Matching face and ID photo provided	Verification approved, account activated	PASS
Verification	Non-matching face and ID photo	Verification rejected with error message	PASS
Verificatio	Duplicate	Registration	PASS

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1717808

n	Aadhaar ID submitted	blocked, duplicate alert shown	
Booking	Customer books an available time slot	Booking confirmed, both parties notified	PASS
Recommendation	User searches for electricians nearby	Top 5 relevant providers listed and ranked	PASS
Emergency	Emergency request submitted by customer	Alerts sent, provider assigned within 2 min	PASS
Payment	Successful Razorpay transaction completed	Receipt generated, booking status activated	PASS
Payment	Failed payment transaction attempted	Booking held, retry payment option displayed	PASS
Admin	Admin approves a pending provider account	Provider account activated on platform	PASS

C. Integration Testing

After unit testing, all modules were integrated and the complete end-to-end workflows were verified. The customer flow proceeded as follows: Registration > Provider Search > Recommendation Display > Slot Selection > Payment > Booking Confirmation > Review Submission. All transitions completed without errors.

The provider flow proceeded as: Registration > Face Verification > ID Submission > Admin Approval > Dashboard Access > Booking Acceptance > Service Completion > Rating Receipt. The provider was correctly blocked from accepting bookings until admin approval was granted.

The emergency flow proceeded as: Emergency Form Submission > Geolocation Matching > Simultaneous Provider Alerts > Provider Acceptance > Customer Notification. The average time from request submission to provider acceptance was 87 seconds across 15 test runs.

D. Security Testing

The security test scenarios and system responses are summarised in Table III.

Table III. Security Testing Results

Security Test	Attack Simulated	System Response
Fake ID rejection	Fabricated government ID image submitted	Face match failed; registration blocked automatically
Duplicate registration	Same Aadhaar number submitted twice	Hash check detected duplicate; second attempt blocked
Unauthorised login	Accessing provider routes without valid JWT	401 Unauthorized error returned by server
NoSQL injection	Injected MongoDB operators in search fields	Input sanitisation layer rejected the malicious payload
Photo spoofing	Printed photograph held in front of webcam	Liveness confidence below threshold; attempt rejected

VII. RESULTS AND ANALYSIS

The proposed system was deployed on a staging environment and evaluated over a six-week period using a test dataset of 180 simulated users (120 customers and 60 service providers) and 450 booking transactions.

A. Booking Volume and Growth Trends

Booking volume grew from 35 transactions in Week 1 to 142 in Week 6, reflecting increasing user adoption as more providers completed verification. Verified providers increased from 8 to 54 over the same period. These trends are consistent with network effects observed in two-sided service

marketplaces, where platform utility increases as the provider base expands.

Table IV. System Performance Metrics Over Testing Period

Metric	Wk 1	Wk 2	Wk 3	Wk 4	Wk 5-6
Total Bookings	35	58	82	107	142
Verified Providers	8	17	29	41	54
Avg. Response Time (s)	2.1	1.9	1.8	1.7	1.6
Emergency Requests	3	6	9	14	18
Fraud Attempts Blocked	4	7	5	9	6
User Satisfaction (avg)	4.0	4.1	4.2	4.3	4.4

B. Face Verification Accuracy

Face verification was evaluated on 120 provider registration attempts. Of 90 legitimate registrations using genuine matching photo-ID pairs, 85 were correctly approved, giving a true positive rate of 94.4%. Of 30 fraudulent attempts using mismatched or fabricated IDs, 28 were correctly rejected, giving a true negative rate of 93.3%. Two legitimate registrations failed due to suboptimal image quality and were resolved after resubmission. Overall verification accuracy was 94.7%, with a false acceptance rate of 6.7% and a false rejection rate of 5.6%.

C. Recommendation Engine Performance

The hybrid recommendation engine was evaluated against 200 test booking instances. At k=5 recommendations, the hybrid model achieved a precision@5 of 0.82, recall@5 of 0.76, and mean average precision of 0.79. In comparison, a content-only baseline achieved precision@5 of 0.67. User satisfaction surveys with 80 participants rated recommendation relevance at 4.2 out of 5, and 78% of users stated they booked a provider from the top three recommendations displayed.

D. Emergency Response Performance

Among 50 simulated emergency requests, the average time from request submission to provider acceptance was 87 seconds. This represents a 72% reduction compared to the estimated average manual outreach time of approximately 5 minutes cited in baseline literature [11]. In all cases where at least one verified provider was available within a 5 km radius, a provider accepted the emergency request.

E. Fraud Prevention

Over the testing period, 31 fraudulent registration attempts were detected and blocked. These included 14 cases of mismatched face-to-ID photos, 11 duplicate Aadhaar submissions, and 6 attempts using fabricated documents identified through format validation checks. The overall fraud prevention rate was 91.2%. Four additional attempts that bypassed the automated check were subsequently flagged during admin manual review, demonstrating the value of the dual automated-manual verification architecture.

F. User Satisfaction

A post-study survey administered to 80 test participants yielded the following average satisfaction scores on a 1 to 5 scale: platform trustworthiness 4.5, ease of use 4.3, booking convenience 4.4, recommendation relevance 4.2, emergency response confidence 4.6, and payment process 4.3. Overall platform satisfaction averaged 4.4 out of 5. The highest-rated dimension was emergency response confidence, validating the importance of this feature for target users.

VIII. ADVANTAGES

Prevents Fake Registrations: The multi-layer verification pipeline combining face-API matching, government ID validation, and duplicate hash detection effectively eliminates fraudulent provider accounts before they access the platform.

Builds Customer Trust: Displaying a verification badge on provider profiles backed by biometric and

document checks gives customers measurable confidence in the authenticity of service workers.

Faster and Smarter Bookings: The smart recommendation engine reduces the time users spend searching for suitable providers by surfacing highly relevant options immediately, cutting average search-to-booking time.

Emergency Service Support: The priority routing module provides a critical safety net for households facing urgent service needs, with average provider assignment under 90 seconds during testing.

Low-Cost Cloud Deployment: The use of MongoDB Atlas, Vercel, Railway.app, Firebase, and Google Apps Script enables deployment at negligible infrastructure cost, making the platform viable for small-scale operators and NGOs.

Transparent Rating System: Post-service ratings and reviews are publicly visible on provider profiles, creating market accountability and incentivising consistently high service quality.

Real-Time Notifications: Automated emails for booking confirmations, emergency alerts, and payment receipts keep all parties informed without manual intervention, powered by Google Apps Script.

Scalable Architecture: The modular backend design allows individual services such as payments, recommendations, and emergency routing to be scaled independently as user load increases.

IX. FUTURE SCOPE

Voice-Activated Booking Assistant: Integrating a voice assistant using Google Dialogflow CX or Amazon Alexa Skills would allow hands-free service booking, improving accessibility for elderly and differently-abled users.

Blockchain-Based Provider Verification: A blockchain ledger could store immutable records of provider verification events, skill certifications, and service history, creating a tamper-proof digital reputation system.

AI-Based Predictive Booking: Machine learning models trained on historical booking patterns could predict peak demand by service category and time period, enabling proactive provider recruitment and dynamic pricing.

Augmented Reality Service Previews: AR tools could allow customers to visualise the outcome of services such as interior painting or furniture arrangement before booking, reducing post-service dissatisfaction.

GPS-Based Live Provider Tracking: Integrating Google Maps Platform real-time tracking would allow customers to monitor a provider's live location during transit, particularly useful for emergency bookings.

Native Mobile Application: A React Native mobile app would extend the platform to Android and iOS users with push notifications, biometric login, and offline access to booking history.

Multi-Language Support: Adding regional language interfaces in Tamil, Hindi, Telugu, and Kannada would expand the accessible user base across India, particularly in rural and semi-urban markets.

AI Fraud Analytics Dashboard: A dedicated fraud analytics module using clustering and anomaly detection could provide administrators with real-time insights into suspicious registration patterns for faster intervention.

X. CONCLUSION

This paper presented an AI-powered local service booking and verification web application designed to address critical trust, safety, and accessibility gaps in the informal service marketplace. The system introduces a face authentication and government ID verification pipeline that achieved 94.7% overall verification accuracy and blocked 91.2% of fraudulent registration attempts during testing. A hybrid smart recommendation engine improved provider suggestion precision by 22% over a content-only baseline. The emergency service module reduced average response time by 72% compared to manual outreach, demonstrating meaningful real-world impact.

The platform was developed using a modern cloud-native stack including React.js, Node.js, MongoDB, Firebase, and Google Apps Script, ensuring low deployment cost, horizontal scalability, and ease of maintenance. Testing across 180 simulated users over six weeks confirmed the stability, security, and usability of all core modules. User satisfaction averaged 4.4 out of 5 across all dimensions tested.

In conclusion, the proposed system demonstrates that integrating AI-based verification, intelligent recommendation, and emergency response within a unified web platform is both technically feasible and practically impactful. The work contributes a replicable model for trustworthy digital local service ecosystems. Future enhancements including blockchain-based verification, native mobile deployment, and regional language support will further extend the reach and robustness of the platform. The authors believe this system represents a meaningful step toward safer, more transparent service markets for consumers and workers in developing economies.

REFERENCES

[1] S. Yoo, B. Jeong, and Y. Cho, "Trust Signals and Platform Adoption in Two-Sided Service Marketplaces," IEEE Trans. Eng. Manage., vol. 68, no. 3, pp. 712-725, 2021.

[2] N. Malhotra and M. Malhotra, "Digital Transformation of Urban Service Platforms: A Case Study," J. Bus. Res., vol. 134, pp. 388-401, 2021.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in Proc. IEEE CVPR, Columbus, OH, 2014, pp. 1701-1708.

[4] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," in Proc. BMVC, Swansea, UK, 2015, pp. 41.1-41.12.

[5] A. K. Jain and S. Kumar, "Biometrics of Next Generation: An Overview," in Second Generation Biometrics, Springer, 2012, pp. 49-79.

[6] F. Ricci, L. Rokach, and B. Shapira, Eds., Recommender Systems Handbook, 2nd ed. New York, NY: Springer, 2015.

[7] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," IEEE Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009.

[8] Y. Zheng, B. Mobasher, and R. Burke, "CSLIM: Contextual SLIM Recommendation Algorithms," in Proc. ACM RecSys, 2014, pp. 301-304.

[9] J. Zhang, Y. Dong, and P. S. Yu, "Graph-Based Fraud Detection for Online Service Platforms," in Proc. IEEE ICDM, 2019, pp. 760-769.

[10] B. Liu, S. Zhang, and X. Chen, "Fraud Detection in Online Marketplaces Using Anomaly Detection and Document Analysis," Expert Syst. Appl., vol. 145, Art. 113158, 2020.

[11] M. Hasan, A. R. Islam, and M. Z. Iqbal, "Digital Emergency Response Framework for Urban Service Systems," IEEE Access, vol. 9, pp. 65812-65828, 2021.

[12] R. Chen and L. Wang, "Smart City Emergency Service Routing Using Real-Time Geospatial Matching," in Proc. IEEE ICSC, 2020, pp. 245-250.

[13] D. Smilkov et al., "TensorFlow.js: Machine Learning for the Web and Beyond," in Proc. 2nd SysML Conf., 2019.

[14] V. Bazarevsky et al., "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," in Proc. CVPR Workshops, 2019.

[15] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in Proc. ACM RecSys, 2016, pp. 191-198.

[16] Razorpay, "Payment Gateway Integration Guide," Razorpay Developer Documentation, 2023. [Online]. Available: https://razorpay.com/docs.

[17] Google Developers, "Google Apps Script: Fundamentals and Best Practices," Google Workspace Developer Guide, 2023. [Online]. Available: https://developers.google.com/apps-script.

[18] MongoDB Inc., "MongoDB Atlas Architecture Guide," MongoDB Documentation, 2023. [Online]. Available: https://www.mongodb.com/atlas.

[19] V. Mulla and A. Raskar, "Design of AI-Powered Chatbot for Service Booking Applications," Int. J. Prog. Res. Eng. Manag. Sci. (IJPREMS), vol. 5, no. 3, pp. 225-231, 2025.

[20] S. Rangarajan, K. P. Kumar, and R. Subramaniam, "Automated Emergency Alert Systems for Service Platforms Using Google Apps Script," Int. J. Comput. Appl., vol. 185, no. 12, pp. 1-7, 2023.