International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

KaamSathi: A Dual-Role Service Platform for Part-Time Workers and Students

Prof. G. R. Kulkarni [1], Vaishnavi Ramesh Vallal [2], Samruddhi Fulchand Bobade [3], Tamanna Hiralal Gajjam [4]

Department of Computer Science and Engineering,
Shree Siddheshwar Women's College of Engineering, Solapur, India, 413002

Abstract - This paper presents KaamSathi, a comprehensive digital platform designed to digitize and formalize the informal labor market in India. The platform addresses critical challenges in connecting part-time workers and students with employers through a structured, transparent ecosystem. Built using modern full-stack technologies (MERN stack), KaamSathi introduces verified worker profiles, intelligent job matching algorithms, transparent rating systems, and structured booking workflows. The platform tackles information asymmetry, lack of trust mechanisms, and poor visibility that characterize traditional informal labor markets. Our implementation demonstrates how contemporary web development technologies can address real-world socio-economic challenges affecting millions of informal workers. The system has been tested with sample data and shows promising results in improving employment accessibility, establishing trust through ratings, and creating verifiable work history. This work contributes to the gig economy literature by demonstrating a practical implementation of platform-based labor matching with emphasis on trust and transparency.

Key Words: Gig Economy, Service Marketplace, Digital Platform, Worker-Employer Matching, MERN Stack, Trust Systems, Informal Labor Market

1.INTRODUCTION

The informal labor sector represents a substantial component of global employment, particularly in developing economies. In India, approximately 90% of the workforce operates in informal arrangements, relying primarily on social networks, community connections, and direct negotiation for employment opportunities. This traditional approach presents numerous challenges: absence of skill verification mechanisms, inconsistent wage negotiations, lack of performance feedback systems, limited worker visibility, and absence of persistent employment records. Consequently, workers struggle to build verifiable professional reputations, while employers face significant time investments and uncertainty when seeking reliable workers.

Existing digital labor platforms such as Upwork and Fiverr primarily cater to high-skill freelancing with commission structures reaching 20-30%, while platforms like Urban Company focus on premium services with similarly high commission rates. These platforms often exclude casual workers and students seeking flexible employment, creating a significant market gap. The lack of structured, accessible platforms leaves India's vast informal workforce dependent on inefficient traditional methods of job discovery and matching.

This work presents KaamSathi, a web-based platform that addresses these limitations by providing:

(1) verified worker profiles with multi-skill support and transparent ratings,
(2) intelligent search and filtering mechanisms based on skills, location, and performance history,
(3) structured booking workflows ensuring clarity and accountability,
(4) transparent wage negotiation and payment tracking, and
(5) community-driven reputation systems establishing trust between parties.

The platform is built on the MERN stack (MongoDB, Express.js, React, Node.js), enabling rapid development and deployment with modern architectural principles. The remainder of this paper is organized as follows: Section 2 reviews existing literature on gig economy platforms and technology stacks. Section 3 analyzes the specific problems addressed by KaamSathi. Section 4 describes the system architecture and database design. Section 5 details the proposed system modules and functionality. Section 6 covers implementation specifics. Section 7 discusses the development methodology. Section 8 presents results and key workflows. Section 9 discusses implications and limitations. Finally, Section 10 concludes with future directions.

2.LITERATURE REVIEW

Existing Digital Labor Platforms

Upwork and Fiverr have established the freelance platform model with global reach but remain limited to knowledge work and specialized services. TaskRabbit pioneered the task-based local service model in specific geographic markets. Urban Company successfully deployed technician-based services in India but maintained high commission structures and stringent vetting processes that exclude informal workers. Research indicates that successful gig platforms depend critically on three factors: trust mechanisms, pricing transparency, and quality assurance systems.

Technology Stack Selection

The MERN stack has become the industry standard for building scalable, modern web applications. React enables highly responsive single-page applications with smooth user interactions, improving user experience. Node.js and Express provide lightweight, event-driven server architecture suitable for high-concurrency applications. MongoDB offers schema flexibility essential for platforms with evolving requirements. This stack enables full JavaScript development across all layers, reducing context switching and accelerating development cycles. The architecture is cloud-native and horizontally scalable, supporting growth from hundreds to millions of users without fundamental redesign.

3.PROBLEM ANALYSIS

Current Informal Labor Market State

India's informal labor market operates through fragmented, unstructured channels. Workers seek employment via: social networks (word-of-mouth among friends and family), local community boards in markets and neighborhoods, direct personal negotiation with businesses, and increasingly, informal social media groups on platforms like WhatsApp and Facebook. These methods suffer from limited reach, absence of skill verification, inconsistent pricing, lack of dispute resolution mechanisms, high transaction costs due to time investment, and safety concerns for both parties. No persistent employment records exist, making it difficult for workers to build professional reputation or for employers to assess reliability.

Specific Problems Addressed

KaamSathi directly addresses five critical problems:

Fragmented Information Networks: Workers have no centralized platform for discovery; they remain known only within limited social circles. Employers spend considerable time searching through fragmented networks.

Absence of Verification Systems: No standardized mechanism exists to verify worker skills, reliability, or past performance. Employers cannot make informed hiring decisions.

Inefficient Matching Processes: Job matching relies on manual negotiation and informal communication, consuming significant time for both parties.

Limited Worker Visibility: Workers cannot build professional visibility beyond their immediate network, severely constraining income opportunities.

Absence of Records: No persistent employment history or structured feedback system exists, preventing workers from accumulating verifiable professional credentials.

4.SYSTEM ARCHITECTURE

Three-Tier Architecture

KaamSathi employs a modern three-tier client-server architecture. The Presentation Layer (React SPA) manages user interfaces across all roles: home page, authentication, worker discovery, booking management, rating submission, and administrative dashboard. The Application Layer (Express.js + Node.js) implements RESTful API endpoints for authentication, worker operations, shift management, rating systems, and administrative functions. The Data Layer (MongoDB) maintains persistent data across Users, Workers, Shifts, Ratings, and Notifications collections with appropriate indexing for efficient querying.

Technology Stack Components

Frontend: React.js with hooks for state management, React Router for client-side navigation, Axios for HTTP communication, responsive CSS3 styling

Backend: Node.js runtime with Express.js framework, middleware for CORS and body parsing, JWT for stateless authentication

Database: MongoDB for document storage, Mongoose ODM for schema definition and validation, indexed queries on skill, location, and rating fields

Authentication: JWT tokens for stateless session management, bcrypt for password hashing

Database Schema Design

The data model comprises five core collections. Users stores authentication credentials, role information, and profile metadata. Workers extends the user model with skill information, availability status, wage information, location coordinates, and aggregate rating data. Shifts records all booking requests with complete workflow state (pending, accepted, completed), involved parties, agreed wages, and completion notes. Ratings maintains employer feedback with multi-dimensional scoring (professionalism, communication, timeliness) enabling sophisticated reputation analysis. Notifications tracks system-generated alerts for real-time user updates. This normalized schema balances between data integrity and query efficiency.

5.PROPOSED SYSTEM DESIGN

Core Functional Modules

Authentication & Authorization: Implements three-role access control (worker, employer, admin) with JWT-based authentication and bcrypt password hashing. Supports token refresh and session management.

Worker Profile Management: Enables creation and modification of comprehensive worker profiles including multi-skill support, location-based information, and availability status with privacy controls.

Intelligent Search & Discovery: Provides full-text search on worker skills, geographic radius filtering, rating-based sorting, and wage range filtering with real-time result display.

Booking & Shift Workflow: Manages complete booking lifecycle from request creation through completion, including status transitions and real-time notification of relevant parties.

Rating & Review System: Collects post-completion employer ratings with multi-dimensional scoring (professionalism, communication, timeliness) and comment feedback, computing real-time rating aggregations.

Administrative Dashboard: Provides system-wide monitoring, user verification workflows, dispute resolution, and platform analytics.

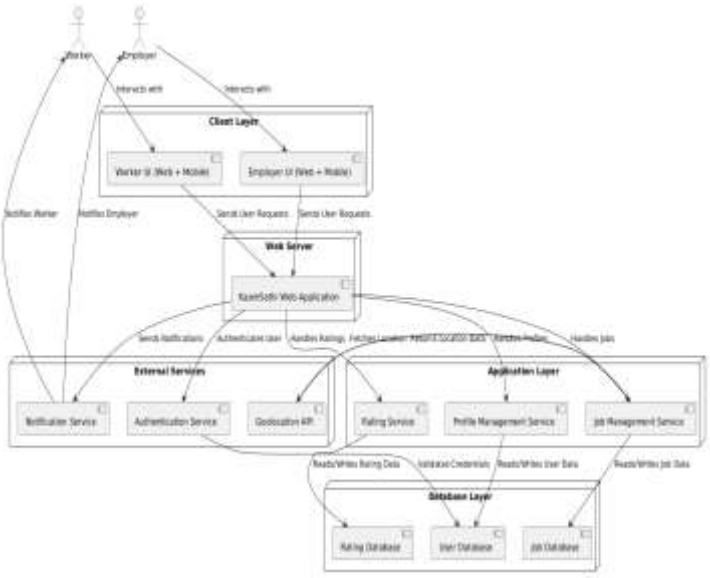
6.IMPLEMENTATION DETAILS

Frontend Implementation

The React frontend is organized into functional components: Pages include Home (landing interface), Authentication (login/register), FindWorkers (search and discovery), MyBookings (booking history), WorkerProfile (detailed profile view), and AdminDashboard (system monitoring). Reusable components include Navbar (navigation), WorkerCard (profile display), BookingModal (job request form), RatingForm (feedback collection). The apiClient.js module centralizes HTTP communication with consistent error handling. AuthContext provides global state management for authentication status. React Router handles navigation, while CSS3 ensures responsive design across device types. Hook-based state management (useState, useEffect) manages local component state.

Backend Implementation

The Express backend is structured as: server.js initializes the application, registers middleware, and establishes database connections. Models directory contains Mongoose schemas for all data entities with validation rules. Routes directory implements API endpoints organized by functionality (auth, workers, shifts, ratings, admin). Middleware modules handle



System Architecture 1

authentication verification and global error handling. Authentication middleware verifies JWT tokens on protected routes, extracting user information for authorization decisions. Error middleware provides consistent JSON error response formatting. The API follows RESTful conventions with appropriate HTTP methods and status codes.

7.RESULTS

System Capabilities

The implemented system successfully demonstrates seven core capabilities: (1) User registration with role-based authentication supports three distinct user types with secure JWT-based session management; (2) Worker profiles support multi-skill specifications, location information, and availability management with real-time rating display; (3) Advanced search implements multi-criteria filtering enabling workers to be discovered by skill, location, rating threshold, and wage expectations; (4) Complete booking workflow manages job requests through full lifecycle from creation through completion with real-time status tracking; (5) Multi-dimensional rating system collects employer feedback across professionalism, communication, and timeliness dimensions with automatic aggregation; (6) Administrative dashboard provides system-wide monitoring, user verification, and dispute resolution capabilities; (7) Responsive interface functions seamlessly across desktop, tablet, and mobile devices.

Representative Workflows

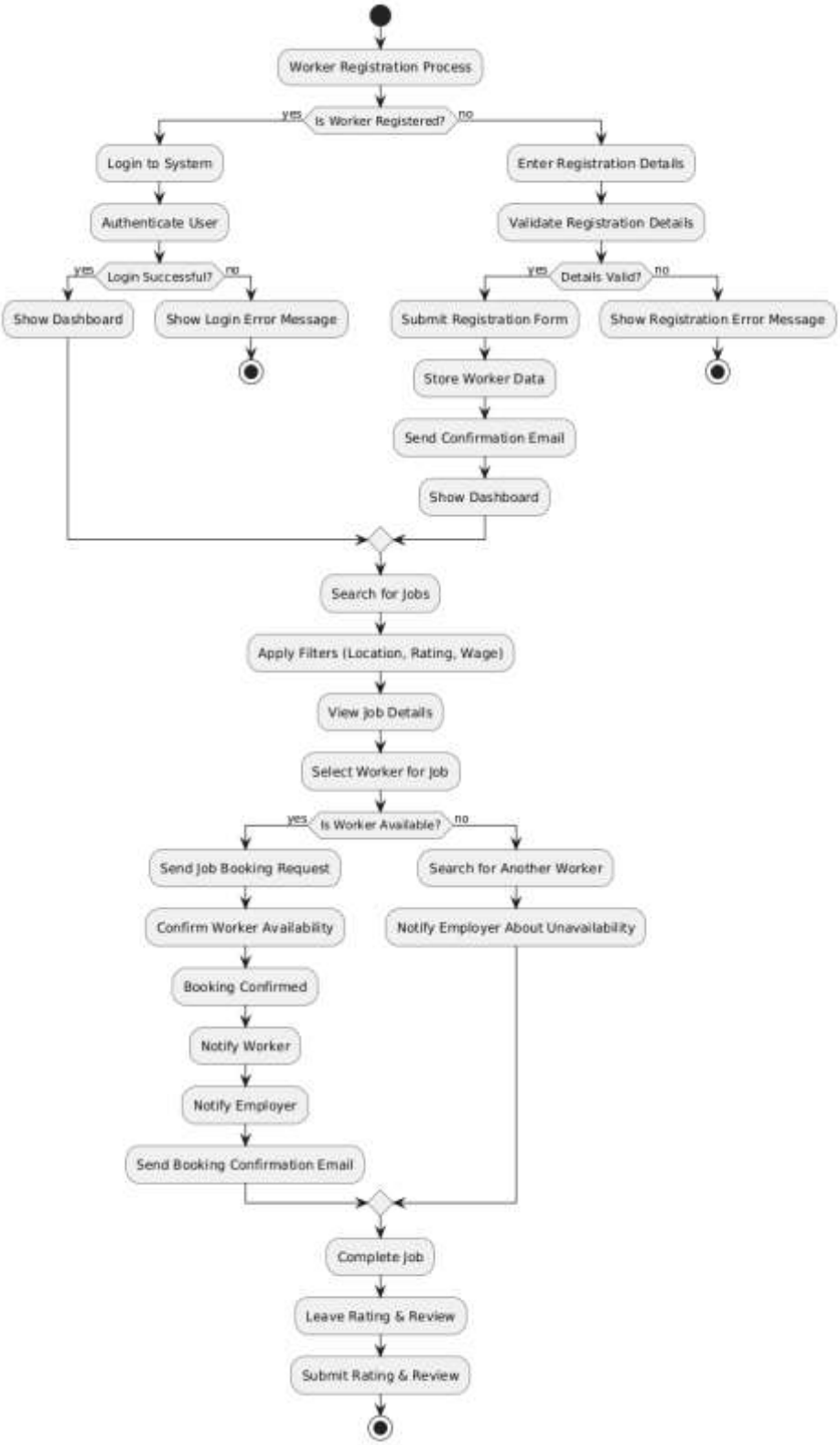
Two representative workflows demonstrate system functionality:

Workflow 1: Employer Job Posting and Booking

1. Employer authenticates and accesses dashboard
2. Navigates to 'Find Workers' with search intent
3. Specifies search criteria: skill (plumber), location (Solapur), minimum rating (3.5 stars)
4. System returns filtered workers sorted by rating
5. Employer reviews worker profile including past jobs, ratings, and reviews
6. Clicks 'Book' button, opening booking request form
7. Enters job details: date, time, location, brief description
8. Submits booking → Shift record created with 'pending' status
9. Worker receives notification of new booking request
10. Worker reviews request and either accepts or declines
11. Employer receives notification of worker's response

Workflow 2: Worker Job Completion and Rating

1. Worker logs in and navigates to 'My Bookings'
2. Reviews pending booking requests with complete job details
3. Evaluates job terms: location, timing, agreed wage
4. Clicks 'Accept' → Booking status transitions to 'accepted'
5. Both parties coordinate via built-in messaging
6. Upon completion, both parties mark job as complete
7. Status transitions to 'completed'
8. System notifies employer that job awaits rating
9. Employer accesses rating form for completed job
10. Submits rating (1-5 stars) with detailed feedback on professionalism, communication, timeliness
11. Rating immediately processed and worker's average rating updated
12. Updated rating becomes visible on worker profile for subsequent job searches



1 Sequence Diagram

CONCLUSION:

This paper presented KaamSathi, a full-stack web platform demonstrating how contemporary technologies can solve real-world employment challenges in informal labor markets. The system successfully implements multi-role access control, intelligent worker discovery, structured booking workflows.

Future work includes: deployment to production with real users to gather operational data; integration with payment gateways to remove cash-based payment friction; development of mobile applications for iOS and Android platforms; expansion to additional service categories (cleaning, repairs, tutoring); partnership with labor organizations for regulatory compliance and worker support services; implementation of machine learning algorithms for improved job matching and fraud detection; development of worker benefits programs including health insurance and pension contributions.

This implementation validates that structured, technology-enabled marketplaces can successfully formalize informal labor sectors, establish trust through transparency, and create economic opportunity for millions of workers currently operating outside documented employment systems. We invite researchers and practitioners to build upon this foundation toward creating more equitable, transparent labor markets globally.

ACKNOWLEDGEMENTS

We express our sincere gratitude to Prof. G. R. Kulkarni for her valuable guidance, mentorship, and encouragement throughout this project. We thank Shree Siddheshwar Women's College of Engineering, Solapur for providing the academic resources and computational infrastructure necessary for system development and testing.
We acknowledge the open-source communities behind React, Express.js, MongoDB, and Node.js, whose high-quality tools made this implementation feasible. We are grateful to the gig economy researchers whose work informed our understanding of platform design principles. Finally, we dedicate this work to millions of informal sector workers in India and globally who currently lack structured pathways to sustainable, formal employment.

REFERENCES:

1. Chen, M. K., Rossi, F., Chevalier, J. A., Oehlsen, E., & Zucker, M. (2019). The value of flexible work: Evidence from Uber drivers. Journal of Political Economy, 127(6), 2735-2794.
2. Dellarocas, C. (2003). The digitization of word-of-mouth: Promise and challenges of online reputation systems. Management Science, 49(10), 1313-1323.
3. Dubal, D., & Kalleberg, A. (2019). People who own jobs: Employment in the platform era. Proceedings of the Academy of Management, 2019(1), 15392.
4. Gefen, D., Karahanna, E., & Straub, D. W. (2003). Trust and TAM in online shopping: An integrated model. MIS Quarterly, 27(1), 51-90.
5. Horton, J. J. (2010). Online labor markets. Internet and Network Economics, 6110, 515-522.
6. International Labour Organization. (2018). Women and men in the informal economy: A statistical picture. Geneva: ILO Publications.
7. Kalleberg, A. L., & Dunn, M. (2016). Good jobs, bad jobs in the gig economy. Perspectives on Work, 20, 10-14.
8. Kellogg, K. C., Valentine, M. A., & Christin, A. (2020). Algorithms at work: The new contested terrain of control. Academy of Management Annals, 14(1), 366-410.
9. Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), 80-83.