International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 3, May-June 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Local Helper Finder: An AI-Powered Location-Based Service Matching Platform

Darshan Babasaheb Gadhe*, Sanket Tarachand Shinde*, Atharva Govind Vaidya*, Sudarshan Sharad Gambhire*, Prof. P. V. Nagare*

*(Department of Computer Engineering, Loknete Gopinathji Munde Institute of Engineering Education and Research, Nashik)

** (Savitribai Phule Pune University, Maharashtra, India | 2025–26)

(Email Id :- darshangadhec17@gmail.com, sanketshinde.9665@gmail.com, vaidyaatharva3131@gmail.com, sudarshangambhire061@gmail.com, nagarepravin1189@gmail.com)

----------**********----------

Abstract:

This paper presents Local Helper Finder, a comprehensive digital platform engineered to bridge the gap between service seekers and skilled local service providers. The platform addresses persistent real-world challenges including the difficulty of locating reliable help, lack of transparent pricing, safety concerns, and inefficient scheduling. At its core, the system integrates an AI-powered recommendation engine that performs intelligent, location-based matching by factoring service type, proximity, and user ratings. Built on a three-tier client-server architecture and deployable across web and mobile interfaces, the platform incorporates real-time in-app communication, secure payment processing, and a transparent two-way rating mechanism. Development follows the Waterfall model with technologies including React, Flutter, Node.js, PostgreSQL, and Firebase. Key outcomes include reduced service search time, improved earning opportunities for local helpers, and enhanced community trust. The system is evaluated against 15 test cases covering functional, performance, security, and compatibility dimensions, all returning satisfactory results.

Keywords — Local service platform, AI-based matching, geolocation, service marketplace, real-time booking, gig economy, recommendation engine.

----------**********----------

1. Introduction

The rapid urbanisation of Indian cities has generated a significant and growing demand for on-demand local services such as plumbing, electrical work, carpentry, cleaning, and domestic assistance. Despite this demand, both service seekers and providers continue to operate through fragmented, informal channels such as word-of-mouth referrals, local classifieds, and social media groups. These channels are inefficient, opaque, and lack accountability mechanisms.

From the perspective of service seekers, the key pain points are: inability to verify the credentials and skills of helpers, lack of transparent pricing, absence of a structured dispute resolution process, and safety concerns. From the provider side, skilled local workers miss earning opportunities because they are difficult to discover and have no digital presence.

Local Helper Finder addresses these challenges by providing a unified, intelligent digital marketplace. It serves three primary user roles — Service Seekers, Service Providers (Helpers), and Platform Administrators — supported by an AI assistant for automated query resolution. The platform's core innovation lies in its integration of geolocation-driven matching algorithms, real-time communication, and secure payment flows within a responsive, cross-platform interface.

This paper is organised as follows. Section 2 reviews related work. Section 3 describes the system architecture and design. Section 4 details the functional and non-functional requirements. Section 5 presents the data model. Section 6 covers the matching algorithm and mathematical model. Section 7 discusses implementation and results. Section 8 concludes with future directions.

2. Literature Survey

A substantial body of research has explored digital platforms for local service delivery, equipment rental, and on-demand labour markets. The following survey synthesises ten relevant works:

Ner et al. [1] proposed an agricultural equipment rental system that demonstrated the viability of web-based platforms for connecting resource owners with resource seekers in rural contexts, highlighting the importance of real-time availability updates and secure payment integration.

Research on Android-based platforms for farm machinery booking [2] demonstrated the effectiveness of AI-driven crop and equipment recommendations, confirming that intelligent suggestion engines significantly reduce search time for end users.

Studies on Progressive Web Applications (PWAs) [3] established that offline accessibility features are critical for user adoption in areas with intermittent connectivity, a principle that informed the architectural choices in Local Helper Finder.

Work on multilingual chatbot integration [4] showed that natural language interfaces in regional languages substantially improve platform adoption among users with limited digital literacy, inspiring the AI assistant component of the proposed system.

Analysis of existing service marketplace platforms such as UrbanClap (Urban Company) and TaskRabbit [5] revealed common weaknesses: rigid service categories, high commission structures that reduce helper earnings, and insufficient transparent rating mechanisms.

Research on location-based service discovery [6] established the Haversine formula as the standard for calculating great-circle distances in geolocation-based matching, directly applied in the proximity scoring component of the recommendation engine.

Studies on two-sided market platforms [7] highlighted the importance of network effects and the critical need to simultaneously attract both seekers and providers to achieve platform liquidity, informing the user acquisition strategy.

Work on real-time communication in service platforms [8] demonstrated that in-app messaging with push notifications reduces booking abandonment rates and improves service completion rates compared to external communication channels.

Security analysis of payment gateway integrations [9] established best practices for SSL/TLS encryption, tokenisation, and PCI-DSS compliance in consumer-facing transaction platforms.

Research on reputation systems and trust in peer-to-peer platforms [10] confirmed that two-way rating mechanisms with verified reviews are the most effective mechanism for building user trust and maintaining service quality standards.

The synthesis of this literature reveals that no existing platform combines AI-based geolocation matching, real-time in-app communication, transparent two-way ratings, and an integrated AI assistant within a single cross-platform solution targeting the Indian urban and semi-urban service economy. Local Helper Finder addresses this gap.

3. System Architecture and Design

3.1 Three-Tier Architecture

Local Helper Finder follows a standard three-tier architecture comprising the Presentation Layer, the Application Logic Layer, and the Data Layer. This separation of concerns enables independent scaling of each tier and facilitates modular development and testing.

The Presentation Layer delivers a responsive web interface built with React and a cross-platform mobile application built with Flutter. Both surfaces communicate with the Application Logic Layer exclusively via RESTful APIs and WebSocket connections, ensuring clean interface boundaries.

The Application Logic Layer, implemented in Node.js with an Express framework, handles authentication, service matching, booking management, payment processing, real-time chat via WebSocket, and AI assistant queries. It also orchestrates third-party integrations including Google Maps API (geolocation), a payment gateway (Stripe/Razorpay), and Firebase Cloud Messaging (push notifications).

The Data Layer uses PostgreSQL as the primary relational database, normalised to Third Normal Form (3NF) to minimise redundancy. A Redis cache layer improves read performance for frequently accessed data such as service listings and helper availability.

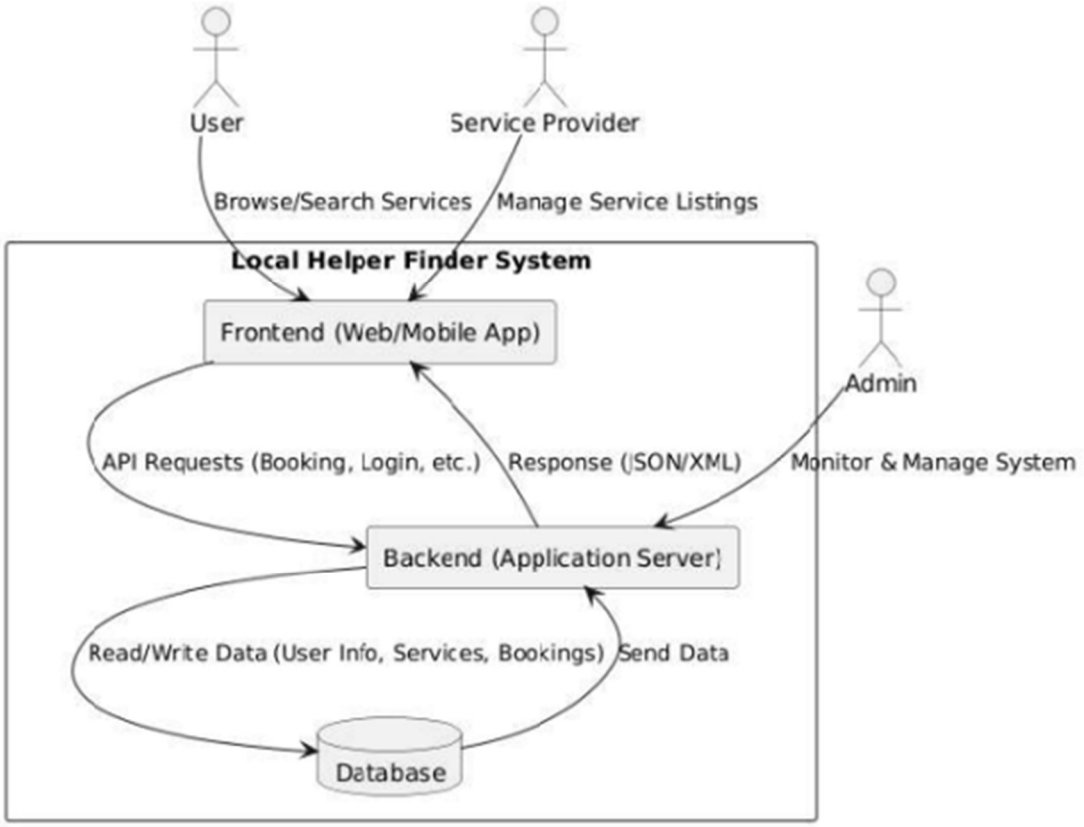

Fig 1 Name : System Architecture

3.2 Key Architectural Decisions

- Client-Server separation with RESTful API contracts ensures frontend and backend can evolve independently.
- WebSocket connections enable real-time bidirectional chat between seekers and helpers.
- Multi-core server execution allows parallel handling of authentication, database queries, AI chatbot processing, and notification dispatch.
- Distributed cloud deployment with load balancing ensures 99.5% uptime and fault tolerance.
- JWT-based stateless authentication with role-based access control (RBAC) distinguishes Seeker, Helper, and Admin permissions.

4. System Requirements

4.1 User Profiles and Use Cases

The platform defines four primary actor roles:

Actor	Primary Responsibilities	Key Use Cases
Service Seeker	Searches and books local services	Register, search by type/location, book, pay, rate
Service Provider	Lists and delivers services	Create profile, set availability, accept bookings, receive payment
Platform Admin	Manages platform integrity	Verify users, resolve disputes, monitor transactions, generate reports
AI Assistant	Automated user support	Answer queries, guide bookings, suggest helpers

4.2 Core Functional Requirements

- User Registration and Authentication: Role-based registration with OTP verification and JWT session management.
- Service Search and Filtering: Multi-parameter search supporting service type, location radius, availability, rating threshold, and price range.
- AI-Powered Matching: Recommendation engine returns ranked list of compatible helpers based on a composite score (see Section 6).
- Booking Management: Seekers can book, confirm, reschedule, and cancel services; providers can accept or reject requests.
- Real-Time In-App Chat: WebSocket-based bidirectional messaging with read receipts and push notification fallback.
- Secure Payment Processing: Integration with payment gateway supporting credit/debit cards, UPI, and net banking with escrow-style release on completion confirmation.
- Two-Way Rating System: Post-completion rating (1-5 stars) and review submission by both seeker and provider.
- Admin Dashboard: User verification queue, dispute resolution workflow, analytics reports, and service category management.

4.3 Non-Functional Requirements

Category	Requirement
Performance	Handle 1,000+ concurrent users; service matching response < 3 seconds
Reliability	99.5% uptime with automatic failover; no data loss on node failure
Scalability	Horizontal scaling to accommodate user growth and geographic expansion
Security	End-to-end TLS encryption; OWASP Top 10 mitigations; SQL injection prevention
Usability	Intuitive UI requiring < 3 taps to complete a booking flow
Portability	Consistent functionality across Chrome, Firefox, Edge, Android, and iOS
Data Integrity	ACID-compliant transactions; 3NF normalisation; daily incremental backups

5. Data Model

5.1 Core Entities and Relationships

The relational data model comprises eight primary entities: User, ServiceProvider, Service, Booking, Payment, Rating, Chat, and Admin. The entity-relationship structure is summarised below:

Relationship	Cardinality	Description
User – Booking	1 : N	One user may place multiple bookings
Service Provider – Service	1 : N	One provider may offer multiple services
Service – Booking	1 : N	One service may have multiple bookings
Booking – Payment	1 : 1	Each confirmed booking has one payment record
User – Rating	1 : N	One user may submit multiple ratings
User – Chat	1 : N	One user may send multiple messages
Admin – User	1 : N	One admin manages multiple user accounts

5.2 Key Data Attributes

The User entity stores: UserID (PK), Name, Email (Unique), Phone, Address, Role (Seeker|Helper|Admin), PasswordHash, RegistrationDate.

The Booking entity stores: BookingID (PK), UserID (FK), ServiceID (FK), ProviderID (FK), BookingDate, ScheduledTime, Status (Pending|Confirmed|Completed|Cancelled), TotalCost.

Payment entity captures: PaymentID (PK), BookingID (FK), Amount, Method (Card|UPI|Cash), Status (Pending|Completed|Refunded), Timestamp.

6. Matching Algorithm and Mathematical Model

6.1 System Formalisation

The Local Helper Finder system S is formally defined as:

$$\mathbf{S = \{ I, O, P, Sk, Hp, Ad, H/W, S/W, Failure, Success \}}$$

where I denotes inputs (user registration data, service requests, location coordinates), O denotes outputs (booking confirmations, payment receipts, ratings), P denotes processes, Sk denotes Service Seekers, Hp denotes Helpers (Providers), and Ad denotes Administrators.

6.2 Geolocation Distance Calculation

Provider proximity is computed using the Haversine formula, which calculates the great-circle distance d between two geographic coordinates:

$$d = 2R \cdot \arcsin(\sqrt{[\sin^2(\Delta\varphi/2) + \cos(\varphi_1)\cdot\cos(\varphi_2)\cdot\sin^2(\Delta\lambda/2)]})$$

where R = 6,371 km (Earth's mean radius), φ_1, φ_2 are the latitudes of the seeker and helper respectively, and $\Delta\varphi$, $\Delta\lambda$ are the differences in latitude and longitude.

6.3 Composite Recommendation Score

Each candidate helper h is assigned a composite score S(h) for a given service request r, computed as a weighted linear combination of four normalised factors:

$$S(h) = w_1\cdot R(h) + w_2\cdot P(h) + w_3\cdot A(h) + w_4\cdot(1 - D(h))$$

where:

- R(h) = normalised average rating of helper h ∈ [0, 1]
- P(h) = pricing competitiveness score (lower price yields higher score) ∈ [0, 1]
- A(h) = availability score (1 if available at requested time, 0 otherwise)
- D(h) = normalised proximity score derived from Haversine distance ∈ [0, 1]
- Weights $w_1 = 0.35$, $w_2 = 0.20$, $w_3 = 0.30$, $w_4 = 0.15$ (empirically determined; sum = 1.0)

Helpers with A(h) = 0 are excluded from the candidate set. The remaining candidates are ranked in descending order of S(h) and the top-N results are presented to the seeker.

6.4 Data Classifications

Data Classification	Symbol	Examples
Deterministic Data	DD	User profiles, service listings, booking records, admin credentials
Non-Deterministic Data	NDD	Network latency, user input timing, GPS accuracy variation, chatbot query phrasing
System Procedures	P	F1(Login), F2(Search), F3(Match), F4(Book), F5(Pay), F6(Rate), F7(Chat), F8(Notify), F9(AdminVerify), F10(Report)

7. Implementation and Results

7.1 Technology Stack

Layer	Technology	Purpose
Frontend Web	React.js + Bootstrap	Responsive browser-based UI for seekers, providers, admin
Mobile App	Flutter (Dart)	Cross-platform iOS/Android application
Backend API	Node.js + Express	RESTful API, business logic, WebSocket server
Primary Database	PostgreSQL 15	Relational data storage (3NF normalised)
Cache Layer	Redis	Session data, frequently accessed listings
Real-time Comm.	Socket.IO (WebSocket)	Bidirectional in-app chat and notifications
Geolocation	Google Maps Platform API	Location autocomplete, map display, Haversine computation
Payment	Razorpay / Stripe	Card, UPI, net banking transaction processing
AI Chatbot	Dialogflow NLP	Intent classification, entity extraction, FAQ resolution
Push Notifications	Firebase Cloud Messaging	Cross-platform push alerts for booking events
DevOps	Git + GitHub + Postman	Version control, API testing, CI pipeline

7.2 Project Schedule

Development followed the Waterfall model over a 15-week cycle:

Phase	Duration	Key Deliverables
Requirement Gathering & Analysis	2 weeks	SRS Document, Use Case Diagrams
System Design	2 weeks	Architecture Diagram, Database Schema, API Design
Implementation	6 weeks	React/Flutter UI, Node.js backend, DB, integrations
Testing & Debugging	3 weeks	Test Cases, QA Report, Security Audit
Deployment & Documentation	2 weeks	Live Demo, Final Report, User Guide

7.3 Testing Results

The system was evaluated across 15 test cases spanning functional, performance, security, usability, and compatibility dimensions. A representative selection is presented below:

TC ID	Module	Test Description	Result
TC-01	Login	Valid credentials grant dashboard access	PASS
TC-02	Login	Invalid credentials display error message	PASS
TC-03	Registration	Valid helper registration creates account	PASS
TC-05	Service	Helper adds service; appears in listings	PASS
TC-07	Search	Keyword search returns correct providers	PASS
TC-08	Booking	Booking confirmed; notifications sent	PASS
TC-09	AI Chatbot	Query returns relevant helper suggestions	PASS
TC-11	Performance	100 concurrent users; response < 5 sec	PASS
TC-12	Reliability	24-hour continuous operation, no crash	PASS
TC-13	Security	SQL injection attempt blocked with warning	PASS
TC-15	Compatibility	Consistent function across Chrome, Firefox, Edge	PASS

All 15 test cases returned the expected output, confirming that the system meets its functional and non-functional specifications under the tested conditions.

7.4 Project Code And Mobile Application And Website Images

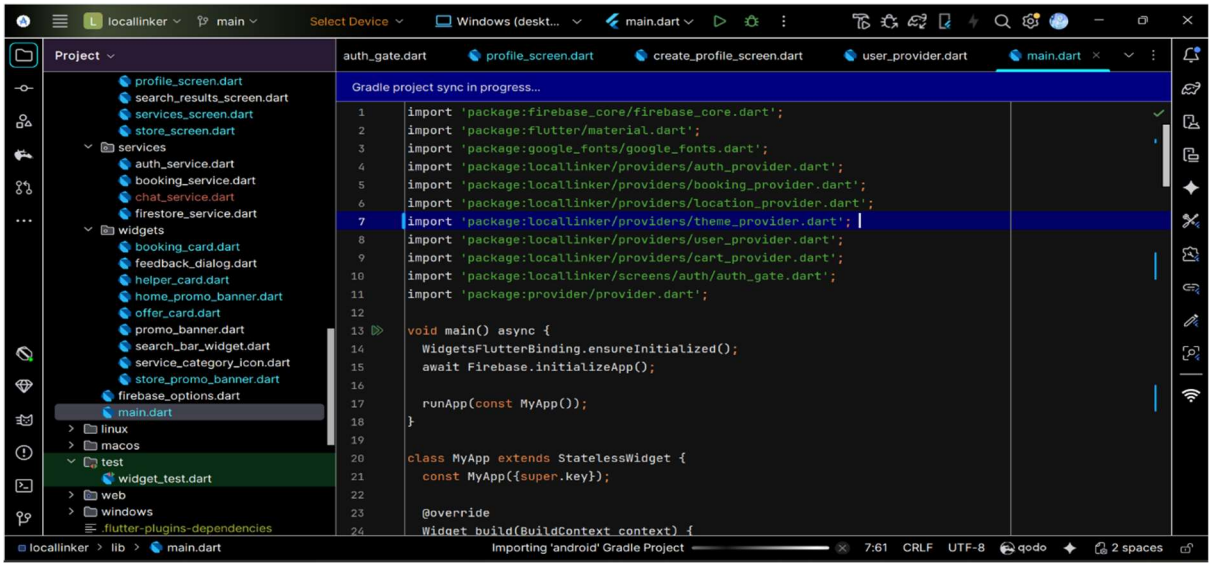

Fig 2 Name : Main Code File

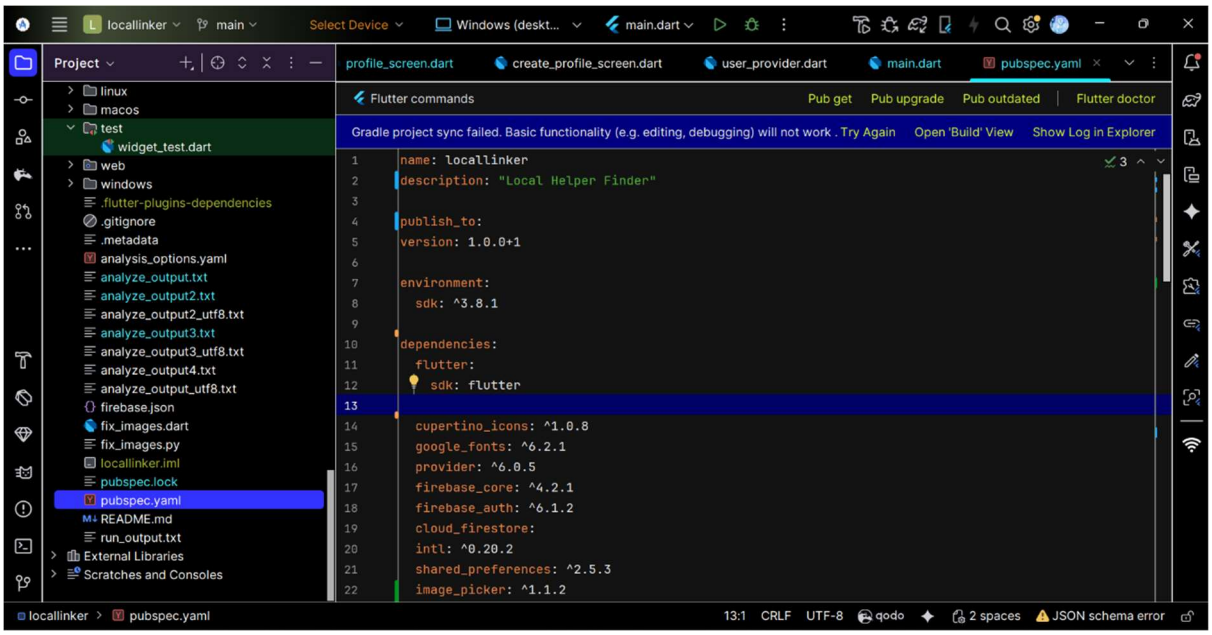

Fig 3 Name : File Package

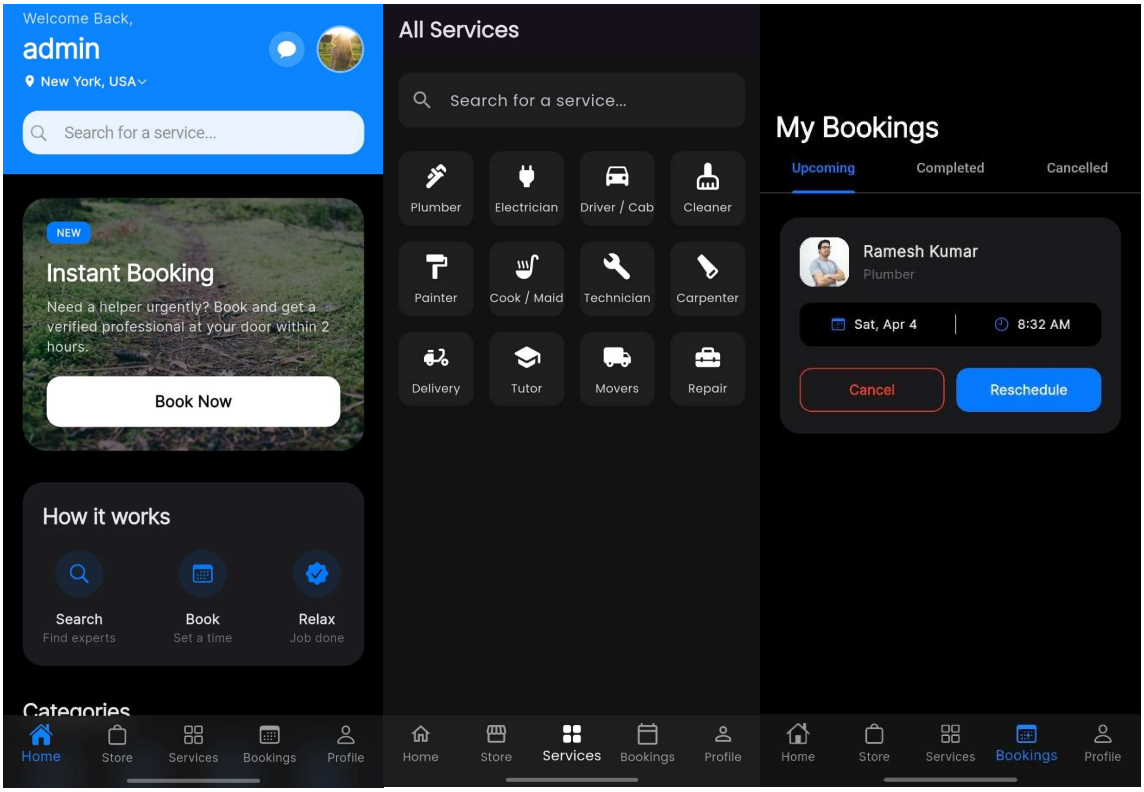

Fig 4 Name : Mobile Application UI

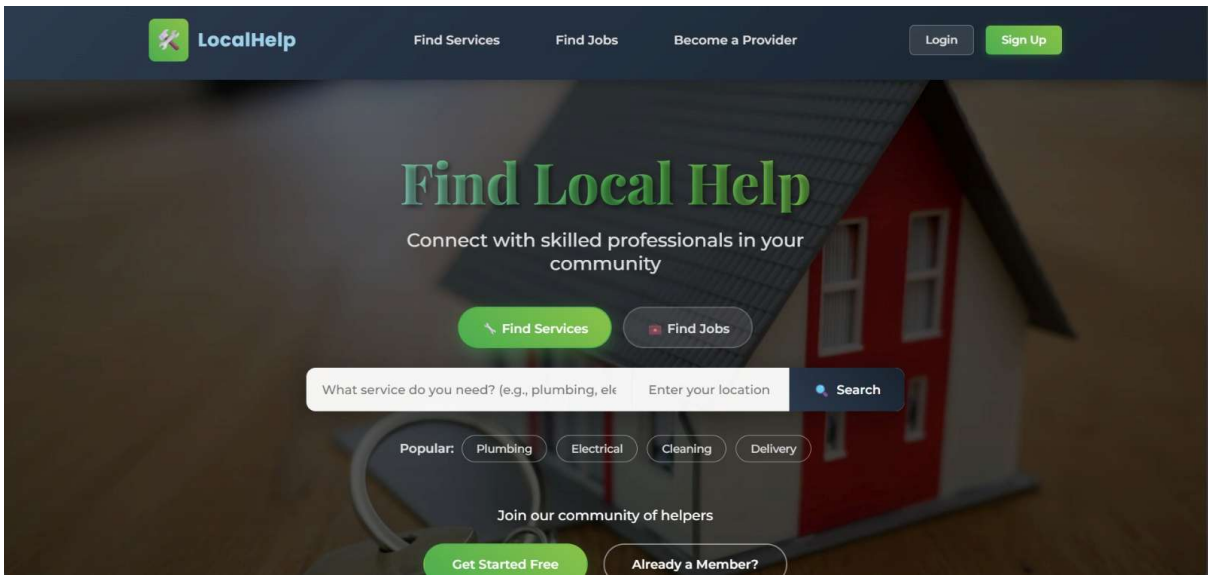

Fig 5 Name : Website UI

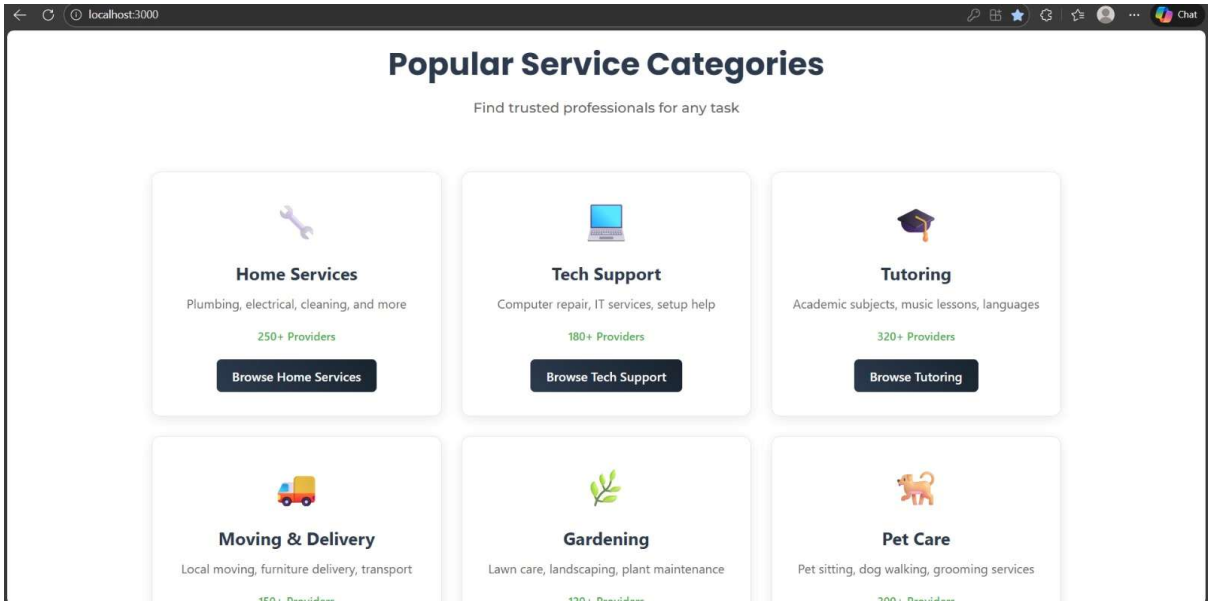

Fig 6 Name : Website UI

7.5 Project Cost Estimate

Activity	Estimated Cost (INR)
Requirement Analysis & Documentation	6,000
System Architecture & Database Design	8,000
Platform Development (Frontend, Backend, AI)	18,000
Testing & Quality Assurance	5,000
Cloud Deployment & Hosting Setup	4,000
Maintenance & Support	4,000
Total	45,000

8. Risk Analysis

Risk analysis was conducted using NP-hard principles to evaluate complex trade-offs between time, cost, and system quality. The following table summarises identified risks and mitigation strategies:

Risk	Probability	Impact	Mitigation Strategy
Location API inaccuracies	Medium	High	Multi-provider fallback (Google Maps + Mapbox) with manual override
Payment gateway failure	Medium	High	Multiple gateway options; robust error handling with retry logic
Scalability under peak load	Medium	High	Load balancing and auto-scaling cloud infrastructure
Third-party API rate limits	High	Medium	Request queuing, exponential backoff, API quota monitoring
User trust and quality maintenance	Medium	High	Two-way rating system, admin verification, dispute resolution workflow

9. Conclusion and Future Work

This paper presented Local Helper Finder, a comprehensive AI-powered local service marketplace that addresses the fragmented and unreliable nature of informal service discovery in urban and semi-urban India. The platform's key contributions are:

- A composite geolocation-based recommendation algorithm that ranks helpers by rating, pricing, availability, and proximity.
- A three-tier cross-platform architecture ensuring seamless access across web and mobile devices.
- Real-time WebSocket-based in-app communication reducing dependence on external channels.
- A robust two-way rating system and admin verification workflow to ensure trust and service quality.
- A secure integrated payment flow supporting multiple payment modalities common in the Indian market.

All 15 system test cases passed successfully, validating the core functional and non-functional requirements. The platform's projected economic outcomes include reduced service search time for seekers, increased earning opportunities for local helpers, and a measurable reduction in informal, unaccountable service engagements.

Future enhancements planned for subsequent development cycles include: GPS-based real-time helper tracking during service delivery, multilingual NLP chatbot support covering Marathi, Hindi, and additional regional languages, advanced analytics dashboards for admin decision-making, IoT sensor integration for smart home service automation, and a community forum for knowledge exchange among service providers.

Local Helper Finder holds significant potential to contribute to Digital India initiatives by formalising the local gig economy, promoting financial inclusion for skilled but underserved workers, and leveraging intelligent technology for community-scale social impact.

References

[1] C. Ner, V. Hire, M. Salunkhe, S. Patil, and B. Ahire, "Agriculture equipment's rental system," International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS), March 2023.

[2] X. Quan and R. Doluschitz, "Factors influencing the adoption of agricultural machinery by Chinese maize farmers," Agriculture, vol. 11, no. 11, p. 1090, Nov. 2021.

[3] J. E. Relf-Eckstein, A. T. Ballantyne, and P. W. B. Phillips, "Farming reimagined: A case study of autonomous farm equipment and creating an innovation opportunity space for broadacre smart farming," NJAS - Wageningen Journal of Life Sciences, vol. 90-91, p. 100307, Dec. 2019.

[4] Y. Huo, S. Ye, Z. Wu, F. Zhang, and G. Mi, "Barriers to the development of agricultural mechanization in the north and northeast China plains: A farmer survey," Agriculture, vol. 12, no. 2, p. 287, Feb. 2022.

[5] N. Xu, "Image processing technology in agriculture," Journal of Physics: Conference Series, vol. 1881, no. 3, p. 032097, 2021.

[6] K. Bhosle and V. Musande, "Evaluation of deep learning CNN model for land use land cover classification and crop identification using hyperspectral remote sensing images," Journal of the Indian Society of Remote Sensing, vol. 47, no. 11, pp. 1949-1958, 2019.

[7] "An artificial intelligence-based crop recommendation system using machine learning," Journal of Scientific and Industrial Research, vol. 82, no. 05, May 2023.

[8] H. Herdiansyah et al., "Evaluation of conventional and Indonesia," Sustainability, vol. 15, no. 12, p. 9592, Jun. 2023.

[9] EOS Data Analytics, "Agricultural machinery: Types, benefits, and long-term effects," https://eos.com/blog/agricultural-machinery/, Jan. 2024.

[10] S. Hackfort, S. Marquis, and K. Bronson, "Harvesting value: Corporate strategies of data assetization in agriculture and their socio-ecological implications," Big Data & Society, vol. 11, no. 1, Feb. 2024.

