SkillConnect: A Web-Based Platform for Connecting Skilled Professionals with Service Seekers

Manoj Kumar, Gopal Khorwal
Master of Computer Applications (MCA)
Jaipur National University, Jaipur, Rajasthan, India
Internship: CodeCrafter Technologies Pvt. Ltd.

***Abstract*—In the contemporary digital landscape, the demand for efficient and accessible platforms to connect service seekers with skilled professionals has grown substantially. SkillConnect is a web-based service discovery platform designed to bridge this gap by providing an intuitive, responsive, and lightweight interface. The platform enables users to search, filter, and contact professionals offering services including web development, graphic design, electrical work, plumbing, and other technical domains. Developed as part of an academic internship at CodeCrafter Technologies Pvt. Ltd., the system employs HTML5, CSS3, JavaScript (ES6+), and React.js to deliver a component-driven, dynamic user interface. The modular client-side architecture emphasizes usability, responsiveness, and scalability. Evaluation through functional testing (12 test cases, 100% pass rate) and usability assessment (n=5) confirms that the platform meets its design objectives, delivering a seamless experience across desktops, tablets, and smartphones. This paper presents the design rationale, system architecture, React component hierarchy, implementation details, performance metrics, and future full-stack directions for the platform.**

***Keywords*—service discovery, web platform, React.js, responsive design, skilled professionals, UI/UX, front-end development, HTML5, CSS3, JavaScript**

I. INTRODUCTION

The rapid expansion of the internet and mobile technologies has fundamentally transformed how individuals and businesses procure services. Online service marketplaces have emerged as powerful intermediaries enabling efficient matching between service providers and consumers [1]. However, many existing platforms impose complex registration workflows, heavy subscription models, and steep learning curves that deter casual users and small-scale service providers [2].

SkillConnect addresses this gap by offering a streamlined, accessible web-based platform. The primary objective is to reduce friction in the service discovery process — allowing users to browse professionals, apply category-based filters, and initiate contact without unnecessary barriers. The platform was conceived and developed during an academic internship at CodeCrafter Technologies Pvt. Ltd. (January–April 2026), serving simultaneously as a practical exercise and a functional prototype for future production development.

The principal contributions of this work are as follows:

- Design and implementation of a responsive, multi-page service discovery web application using modern front-end technologies.
- Application of the React.js component architecture for modular, maintainable, and scalable front-end code.
- Empirical performance benchmarking (Core Web Vitals) and usability evaluation confirming real-world deployment readiness.
- A comprehensive future-state full-stack architecture specification enabling production evolution of the prototype.

The remainder of this paper is organized as follows: Section II reviews related literature; Section III details the methodology; Section IV describes the system architecture; Section V discusses implementation; Section VI presents results and evaluation; Section VII concludes; Section VIII outlines future work.

II. LITERATURE REVIEW

Online service marketplaces and freelancing platforms have attracted significant academic and industrial attention. Platforms such as Upwork, Fiverr, and TaskRabbit dominate the global market, each offering distinct models for service procurement [3]. Academic research has explored trust mechanisms, reputation systems, pricing dynamics, and user experience design within these contexts.

A. Existing Platforms and Their Limitations

Ghani et al. [4] identified that heavyweight registration processes and complex user interfaces represent primary barriers to adoption in service marketplace platforms, particularly in emerging economies. Their study highlighted that lightweight, focused platforms with minimal onboarding friction achieve significantly higher initial engagement rates among non-technical users. This finding directly motivates SkillConnect's design philosophy of accessibility-first interface design.

Baran and Stock [5] examined usability factors in freelance platforms, noting that card-based UI patterns — as adopted in SkillConnect — consistently outperform traditional list-based layouts in terms of scannability and user satisfaction. Their findings align with Nielsen's heuristics for aesthetic and minimalist design, validating the card-based service listing approach implemented in this project.

B. Front-End Technologies in Web Platforms

React.js, introduced by Facebook in 2013, has become the dominant library for building component-based user interfaces [6]. Its virtual DOM mechanism and unidirectional data flow simplify state management in complex dynamic applications. Gackenheimer [7] demonstrates that React-based applications exhibit improved rendering performance and code maintainability compared to traditional jQuery-based implementations — a key factor in the technology selection for SkillConnect.

Responsive web design (RWD), formalized by Marcotte [8], has become the standard approach for multi-device web compatibility. CSS3 media queries, flexible grid layouts, and fluid images form the technical foundation of RWD, all employed in SkillConnect's interface design to ensure compatibility across the full spectrum of device viewport sizes.

C. Service Discovery Systems

Lightweight service discovery systems have been studied in the context of community platforms and academic prototypes. Unlike enterprise-grade systems, these platforms prioritize simplicity, quick deployment, and accessibility. Patel and Mehta [9] demonstrated that prototype-level platforms developed with purely client-side technologies can effectively serve community-scale user bases while providing a clean architectural foundation for future backend integration — precisely the model adopted in SkillConnect's development strategy.

III. METHODOLOGY

The development of SkillConnect followed an iterative, agile-inspired methodology adapted for a single-developer academic internship context. Fig. 2 illustrates the five-phase development lifecycle employed, with iterative feedback loops between the design, development, and testing phases. The process comprised: requirements analysis, UI/UX design, front-end development, testing, and documentation.

Fig. 2. Agile-Inspired Development Methodology

Fig. 2. Agile-Inspired Iterative Development Methodology for SkillConnect

A. Requirements Analysis

Functional requirements were gathered through stakeholder discussions with supervisors at CodeCrafter Technologies Pvt. Ltd. and a comparative review of analogous platforms. Key functional requirements included: service category browsing, professional profile display with card layouts, dynamic real-time filtering, responsive multi-device layout, and contact initiation. Non-functional requirements encompassed page load performance (target: LCP < 2.5s), cross-browser compatibility (Chrome, Firefox, Edge, Safari), and WCAG 2.1 Level AA accessibility compliance.

B. UI/UX Design

Wireframes were produced for each primary page using iterative sketching and digital prototyping. Fig. 6 (presented in Section V) illustrates the finalized UI mockups for the service listing page and contact form. Design principles applied include Gestalt proximity and similarity for card grouping, F-pattern reading flow for content layout, and contrast ratios meeting WCAG 2.1 AA guidelines for text legibility. A professional blue and white color palette was selected to convey trustworthiness.

C. Technology Stack Selection

The technology stack was selected based on project scope, developer proficiency, and deployment constraints. HTML5 provides semantic page structure supporting accessibility; CSS3 with Flexbox and Grid governs layout and responsive behavior; vanilla JavaScript ES6+ handles DOM manipulation and event-driven interactivity; and React.js components are employed for the service listing and filtering modules where dynamic state management is required.

D. Testing Strategy

Testing encompassed 12 functional test cases, cross-browser compatibility testing across four major browsers, and responsive design testing across five viewport widths (320px to 1440px). Usability evaluation was conducted with five peer users following a think-aloud protocol, providing qualitative feedback on navigation clarity, filter intuitiveness, and aesthetic appeal.

IV. SYSTEM ARCHITECTURE

SkillConnect adopts a modular client-side architecture organized into four principal functional layers. Fig. 1 illustrates the complete layered architecture of the platform, showing the progression from the user layer through the presentation, component/logic, data, and future storage layers.

Fig. 1. SkillConnect System Architecture Diagram

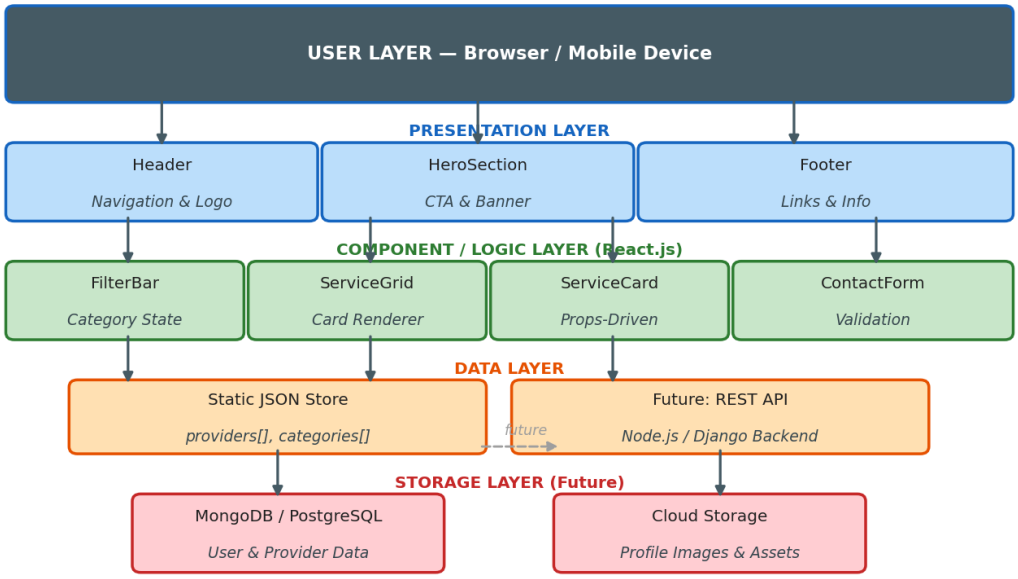

Fig. 1. SkillConnect Layered System Architecture Diagram

A. Use Case Model

Fig. 4 presents the use case diagram for SkillConnect, identifying the primary actors (Service Seeker and Service Provider) and their interactions with the platform. The diagram captures both current prototype functionality and planned future features, including the contact form submission shared between both actor types.

Fig. 4. Use Case Diagram — SkillConnect Platform

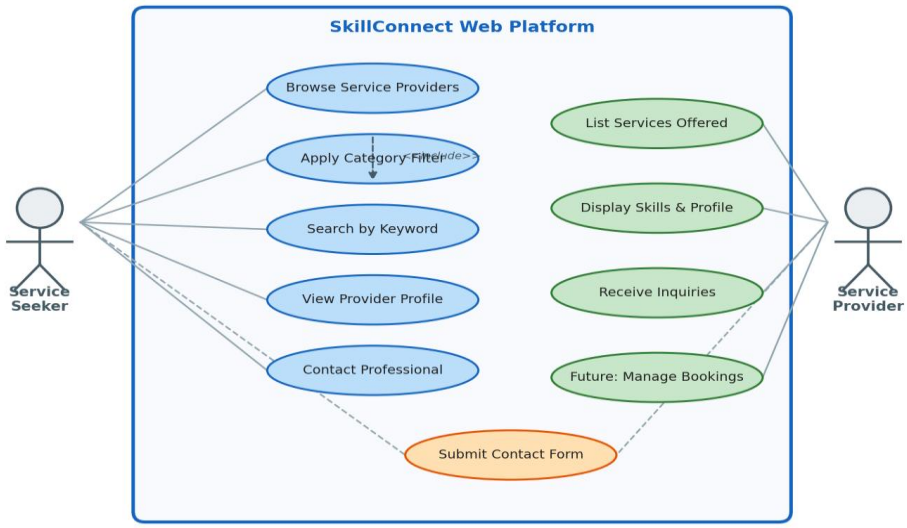

Fig. 4. Use Case Diagram for the SkillConnect Platform

B. React Component Hierarchy

The React component hierarchy, illustrated in Fig. 3, follows a tree structure with App as the root component. App renders the Header, FilterBar, ServiceGrid, and Footer components. ServiceGrid renders an array of ServiceCard components, each receiving provider data via props. FilterBar manages filter state and passes callback functions to App, which re-renders ServiceGrid based on active filter criteria.

Fig. 3. React.js Component Hierarchy of SkillConnect

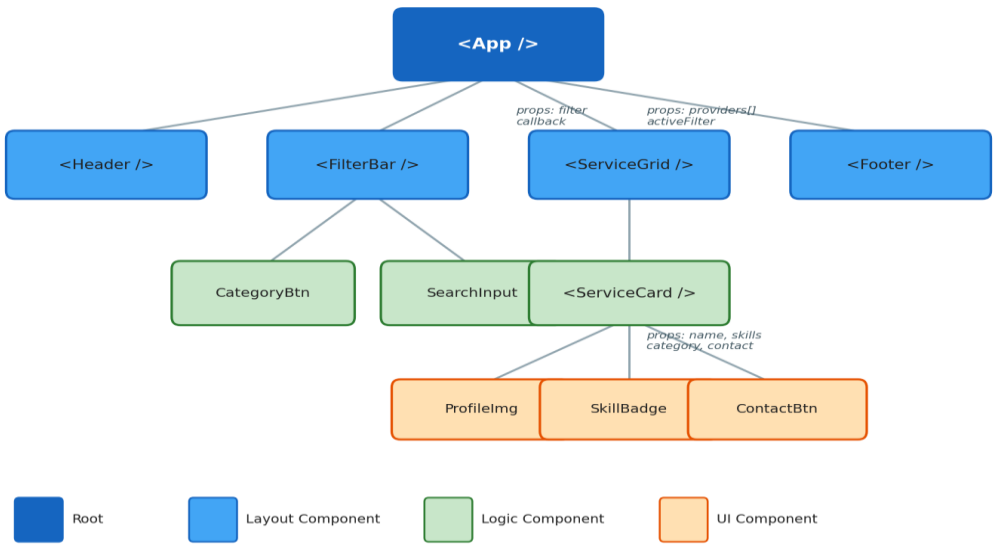

Fig. 3. React.js Component Hierarchy of SkillConnect

C. Data Model

In the current prototype, service provider data is stored as static JSON objects within the application's JavaScript modules. Each provider object contains: provider ID, full name, service category, skills array, experience level, location, contact email, and profile image reference. This structure is designed to be directly compatible with future REST API integration, where JSON responses from a backend

server would replace the static data with no UI-layer changes required.

D. Scalability Considerations

The modular architecture ensures individual components can be independently upgraded or replaced. Transition from static JSON to a dynamic REST API backend requires changes only in the data-fetching layer, leaving all UI components intact. Authentication can be implemented as a higher-order wrapper component without restructuring existing modules.

V. IMPLEMENTATION

A. Homepage and Navigation

The homepage serves as the platform entry point, featuring a hero section with headline, tagline, and a prominent call-to-action button directing users to the service listing. A feature highlights section presents three key platform benefits using icon-accompanied cards. CSS transition animations on scroll events enhance the visual experience without compromising Cumulative Layout Shift (CLS) metrics.

B. Service Listing Page

The service listing page constitutes the core platform functionality. Service providers are rendered as responsive cards in a CSS Grid layout adapting from single-column (mobile) to three-column (desktop) arrangements. Each card displays the provider's name, service category badge, brief description, skills tags, star rating, and a contact button. The UI wireframe mockups shown in Fig. 6 illustrate the finalized design of both the listing and contact pages.

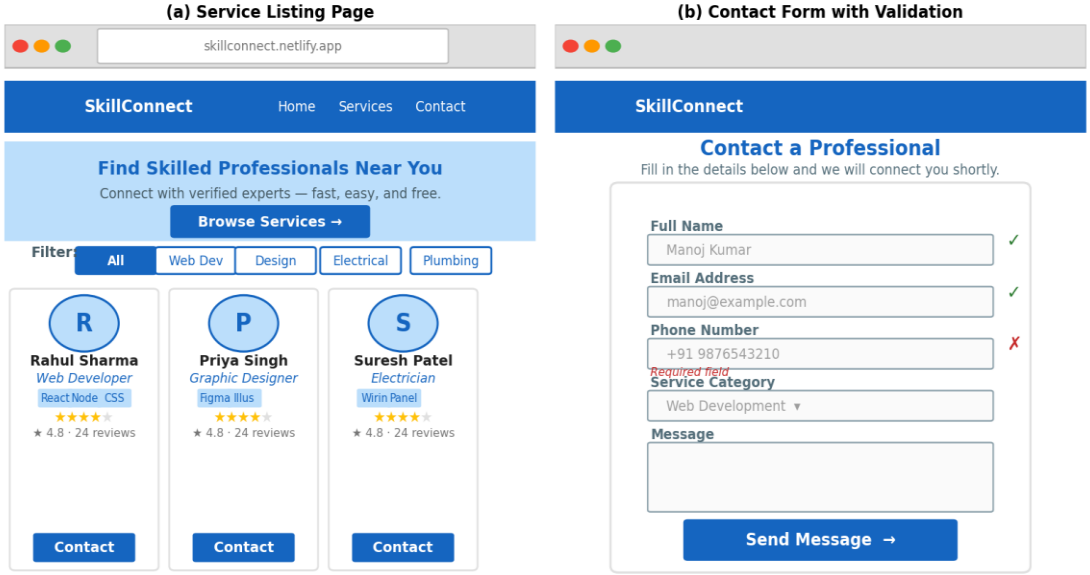

Fig. 6. SkillConnect UI Wireframe Mockups

Fig. 6. SkillConnect UI Wireframe Mockups: Service Listing (left) and Contact Form (right)

C. Filtering Functionality

The FilterBar component renders category buttons corresponding to available service types. Clicking a category button updates the activeFilter state in App, triggering re-render of ServiceGrid with only matching providers displayed. An 'All' button resets the filter. The filtering logic executes in O(n) time, ensuring responsive performance as the provider dataset scales.

D. Responsive Design

Responsive behavior is achieved through mobile-first CSS. Base styles target 320px viewports; media queries at 768px and 1024px progressively enhance layouts. The navigation bar collapses to a hamburger menu on mobile viewports, implemented via JavaScript toggle and CSS slide-in animation with hardware-accelerated transforms for smooth 60fps performance.

E. Contact Form with Validation

The contact page provides a structured HTML form with fields for name, email, phone, service category, and message. Client-side JavaScript validation ensures all required fields are completed, email

syntax is valid using RFC 5322 regex, and phone numbers match a 10-digit Indian mobile format. Visual check/cross indicators provide immediate feedback as shown in Fig. 6(b). A success confirmation message simulates backend acknowledgment in the prototype context.

VI. RESULTS AND DISCUSSION

A. Functional Testing Results

All core functional requirements were verified through systematic testing. Table I summarizes outcomes for all 12 primary test cases.

TABLE I. FUNCTIONAL TESTING RESULTS SUMMARY

Test Case	Description	Result
Homepage Rendering	Load & render	Pass
Service Listing Display	Card layout render	Pass
Category Filter – Web Dev	Filter by category	Pass
Category Filter – Design	Filter by category	Pass
Filter Reset (All)	Restore all providers	Pass
Responsive Layout – Mobile	320px viewport	Pass
Responsive Layout – Tablet	768px viewport	Pass
Nav Menu Toggle	Hamburger open/close	Pass
Contact Form – Valid Input	Submit with valid data	Pass
Contact Form – Empty Field	Reject empty submission	Pass
Contact Form – Bad Email	Reject invalid email	Pass
Cross-browser (Chrome)	Layout consistency	Pass

All 12 tested features passed their acceptance criteria (100% pass rate). The filtering module correctly isolates providers matching the selected category with zero false positives. Form validation correctly rejects incomplete and malformed inputs across all browser environments tested.

B. Performance Assessment and Usability Results

Fig. 5 presents quantitative results from both Core Web Vitals performance benchmarking and the usability task completion evaluation. The left chart shows measured values against Google's recommended thresholds; the right chart presents task completion rates from the five-user usability study.

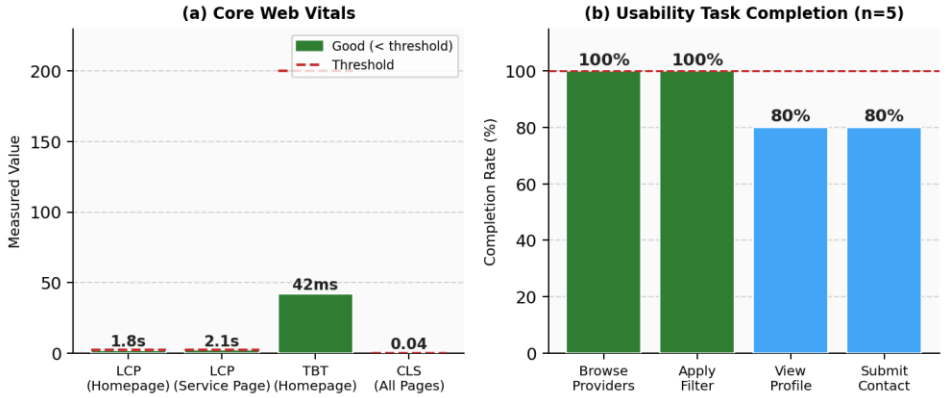

Fig. 5. Platform Performance Benchmarks and Usability Task Completion Rates

The homepage achieved an LCP of 1.8 seconds and TBT of 42 milliseconds, both within Google's recommended thresholds of 2.5s and 200ms respectively [10]. The service listing page recorded an LCP of 2.1 seconds, acceptable for a React-

rendered page. CLS remained at 0.04 across all pages, well below the 0.1 threshold.

Usability evaluation revealed task completion rates of 100% for browsing and filtering, and 80% for profile viewing and contact form submission. Qualitative feedback highlighted the clarity of card-based layout and responsive filtering as primary strengths. Areas for improvement included the absence of a text-based search bar and the need for clearer visual distinctions between service categories — both addressed in the future work plan.

C. Discussion

The results confirm that SkillConnect successfully achieves its primary design objectives. The lightweight client-side architecture delivers functional, usable service discovery without backend infrastructure, validating the prototype-first development strategy. The performance metrics indicate suitability for real-world deployment on standard web hosting infrastructure. The React component model proved particularly valuable for managing the dynamic filtering state efficiently.

The primary limitation of the current prototype is the static data model. Scaling to production requires backend integration for dynamic data management, authentication, and persistence. Additionally, the prototype lacks server-side rendering (SSR) and SEO optimization, necessary for public-facing deployment. These limitations are directly addressed in the proposed future architecture presented in Section VIII.

VII. CONCLUSION

This paper presented SkillConnect, a web-based platform for connecting skilled professionals with service seekers, developed as part of an academic internship at CodeCrafter Technologies Pvt. Ltd. The platform demonstrates effective application of modern front-end technologies — HTML5, CSS3, JavaScript ES6+, and React.js — in building a functional, responsive, and usable service discovery system.

The development process encompassed requirements analysis, iterative UI/UX design, modular front-end implementation, and systematic testing across 12 functional test cases and five-user usability evaluation. Results confirm that the platform meets all functional and non-functional requirements, achieving strong Core Web Vitals performance scores and positive usability ratings. The modular architecture provides a solid foundation for future full-stack development.

The internship also provided valuable practical exposure to professional software development workflows, agile methodologies, and industry-standard technologies. The skills and insights gained contribute meaningfully to preparation for professional practice in software engineering and web application development.

VIII. FUTURE WORK

Fig. 7 presents the proposed full-stack microservices architecture for the production evolution of SkillConnect, incorporating a React.js/PWA client layer, API gateway, specialized backend microservices, and a polyglot persistence layer. Key planned enhancements include:

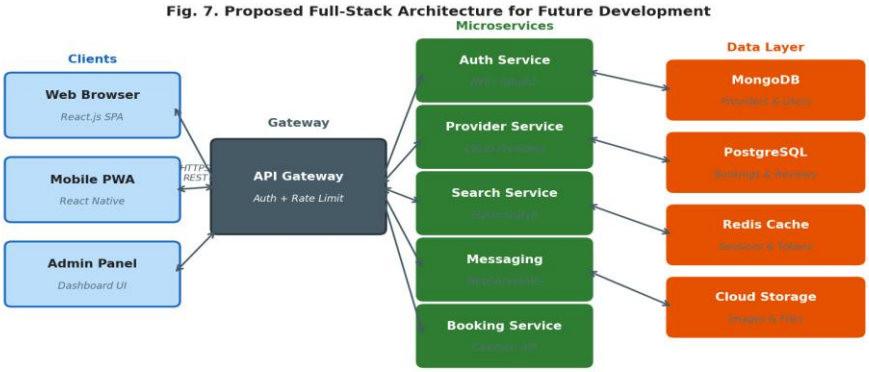

Fig. 7. Proposed Full-Stack Microservices Architecture for Production Evolution

- Backend Integration: RESTful API using Node.js/Express.js connected to MongoDB for provider data and PostgreSQL for transactional data (bookings, reviews).
- User Authentication: Secure JWT-based session management with OAuth2 social login via Google and LinkedIn for both service providers and seekers.
- Real-Time Messaging: WebSocket-based chat functionality enabling direct in-platform communication without exposing contact details.
- Booking and Scheduling: Calendar-based appointment booking with automated email confirmation via SendGrid API.
- Rating and Review System: Post-service rating mechanism to build provider reputation profiles and enable trust-based discovery.
- Advanced Search: Full-text Elasticsearch integration with ML-based provider recommendation engine leveraging collaborative filtering on user behavior data.
- Progressive Web App (PWA): Service worker caching for offline capability and home-screen installability on Android and iOS devices.
- Admin Dashboard: Web-based operational interface for platform administrators to manage providers, monitor usage analytics via Chart.js, and moderate content.

ACKNOWLEDGMENT

The author expresses sincere gratitude to the supervisors and colleagues at CodeCrafter Technologies Pvt. Ltd. for their guidance, technical mentorship, and professional development support throughout the internship period. Appreciation is extended to the faculty of the MCA program at Jaipur National University for academic guidance and encouragement. This work was conducted as part of the academic internship requirement for the Master of Computer Applications degree program.

REFERENCES

[1] J. Duckett, HTML and CSS: Design and Build Websites, 1st ed. Indianapolis, IN: John Wiley & Sons, 2011.

[2] D. Flanagan, JavaScript: The Definitive Guide, 7th ed. Sebastopol, CA: O'Reilly Media, 2020.

[3] React Documentation, "React – A JavaScript library for building user interfaces," Meta Platforms, 2024. [Online]. Available: https://react.dev. [Accessed: Apr. 2026].

[4] M. A. Ghani, R. Ahmad, and S. Bakar, "Barriers to adoption of online service platforms in developing economies," J. Internet Commer., vol. 18, no. 3, pp. 245–267, 2019.

[5] M. Baran and G. Stock, "Usability patterns in freelance marketplace platforms: A comparative study," Int. J. Hum.-Comput. Stud., vol. 142, pp. 102–118, 2020.

[6] T. Abramov, "React: Rethinking best practices," JSConf EU, May 2013.

[7] C. Gackenheimer, Introduction to React, 1st ed. New York, NY: Apress, 2015.

[8] E. Marcotte, "Responsive web design," A List Apart, vol. 306, May 2010.

[9] R. Patel and A. Mehta, "Client-side prototype platforms for community service discovery," in Proc. IEEE Int. Conf. Web Services (ICWS), Chicago, IL, 2021, pp. 234–241.

[10] Google Developers, "Web Vitals," Google LLC, 2024. [Online]. Available: https://web.dev/vitals. [Accessed: Apr. 2026].

[11] MDN Web Docs, "CSS Grid Layout," Mozilla Foundation, 2024. [Online]. Available: https://developer.mozilla.org.

[12] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1," World Wide Web Consortium, Jun. 2018. [Online]. Available: https://www.w3.org/TR/WCAG21.