IJVAST | Vol. 1, Issue 4 | April 2026 | Open Access — Peer Reviewed Research Paper

# AutonationIndia: Design and Development of a High-Performance Interactive Frontend for an Indian Used-Car Marketplace Platform Using React.js, Progressive Web App Architecture, and AI-Assisted Price Estimation

Akanksha Gulave[1], Hema Darne[2]

*Department of Computer Science*

*Dr. Moonje Institute of Management and Computer Studies, Nashik, Maharashtra, India*

***Abstract***—India's used-car market, valued at USD 32 billion in 2024 and projected to reach USD 74 billion by 2030, is characterised by significant information asymmetry between buyers and sellers, fragmented listing platforms, and poor user experience on existing digital marketplaces — particularly on mobile devices and in low-bandwidth environments prevalent in Tier-2 and Tier-3 Indian cities. This paper presents AutonationIndia, a high-performance, mobile-first interactive frontend for an Indian used-car marketplace platform, designed and developed at Dr. Moonje Institute of Management and Computer Studies, Nashik. The platform is built on React.js 18 with TypeScript, implementing Progressive Web App (PWA) architecture for offline capability and installability. Key innovations include: (1) a 15-parameter advanced filtering engine with real-time result updates via Zustand state management; (2) an AI-assisted price estimation module using a gradient-boosted regression model served via REST API, providing instant fair-price ranges for 18,400 car model-year-variant combinations; (3) a Three.js powered 360° virtual car inspection viewer; (4) a live EMI calculator integrated with RBI-updated bank lending rates; and (5) a four-car side-by-side specification comparison tool. The interface is localised for Marathi and Hindi in addition to English using react-i18n. Comprehensive performance evaluation using Google Lighthouse demonstrates scores of 94 (Performance), 98 (Accessibility), 96 (Best Practices), and 100 (SEO). Core Web Vitals targets are met with FCP of 1.2 s, LCP of 2.1 s, TBT of 48 ms, and CLS of 0.04. User testing with 42 participants from the Nashik region achieves a System Usability Scale (SUS) score of 86.4, classified as 'Excellent'. The platform source code and design system are released as open-source.

***Index Terms***—React.js, Progressive Web App, Car Marketplace, Frontend Development, UI/UX Design, Core Web Vitals, Lighthouse, Three.js, AI Price Estimation, Zustand, TypeScript, PWA, Used Car Market India, System Usability Scale, WCAG Accessibility, Tailwind CSS.

## I. INTRODUCTION

India's automotive sector is undergoing a structural transition that places the used-car segment at the centre of digital disruption. With new car sales facing headwinds from rising fuel prices and economic uncertainty, the used-car market has emerged as the primary vehicle ownership pathway for India's growing middle class — particularly in Tier-2 and Tier-3 cities like Nashik, Nagpur, Aurangabad, and Pune. A landmark study by KPMG India [1] found that 62% of first-time car buyers in non-metropolitan cities now prefer a certified pre-owned vehicle over a new purchase, citing lower total cost of ownership as the primary driver.

Despite this growth trajectory, the digital experience offered by existing Indian used-car platforms falls significantly short of international benchmarks. A systematic evaluation of five major platforms — CarDekho, Cars24, OLX Autos, CarWale, and Spinny — conducted as part of this research revealed consistent deficiencies: average Lighthouse performance scores below 78, First Contentful Paint exceeding 2.4 seconds on 4G connections, absence of offline capability,

### *C. Three.js 360° Virtual Car Inspector*

The 360° virtual inspector renders a three-dimensional representation of the selected car using Three.js r158 with WebGL rendering. Car models are represented as textured GLTF 2.0 meshes, with each car make-body-type combination mapped to one of 48 base mesh templates. Seller-uploaded photos are projected onto the mesh using UV mapping to approximate the actual car's appearance. The viewer implements OrbitControls for mouse and touch-based rotation, PerspectiveCamera with 75° field of view, and HDR environment lighting using an HDRI environment map to provide realistic reflections on car surfaces. Hotspot annotations (clickable areas on the 3D model) highlight seller-noted damage points or notable features. The inspector degrades gracefully to a standard image gallery carousel on devices without WebGL support [16].

### *D. Live EMI Calculator*

The EMI calculator fetches current car loan interest rates from the Reserve Bank of India's DBIE (Database on Indian Economy) public API and aggregates rates from eight major

limited regional language support, and filtering systems incapable of simultaneous multi-parameter queries [2]. These deficiencies disproportionately impact users in bandwidth-constrained environments and non-English-speaking demographics — precisely the user segments that constitute the next growth wave for Indian automotive e-commerce.

AutonationIndia is designed to address these deficiencies through a principled application of modern frontend engineering practices: component-based architecture in React.js 18 with concurrent rendering, PWA service workers for offline listing access, code splitting and lazy loading for performance, and an accessibility-first design system aligned with WCAG 2.1 Level AA. The platform is explicitly designed for the Nashik regional context — supporting Marathi as a primary UI language, pricing in Indian Rupees with lakh/crore notation, and car inventory reflective of regional brand preferences (Maruti Suzuki, Tata, Mahindra, Hyundai) [3].

The core research contributions of this paper are: (1) the design, implementation, and open-source release of the AutonationIndia frontend; (2) a systematic feature and performance comparison against five major incumbent platforms; (3) the AI price estimation module architecture and its integration into the user experience; (4) the 360° Three.js viewer implementation for used-car inspection; and (5) user testing results with 42 Nashik-region participants establishing baseline usability benchmarks for regional Indian automotive marketplace UX.

## II. RELATED WORK

### *A. Digital Automotive Marketplaces*

Online used-car marketplaces emerged in the early 2000s with platforms like AutoTrader (US) and eBay Motors demonstrating that digital channels could effectively facilitate high-value automotive transactions previously confined to physical dealerships. Subramanian and Hariharan [4] reviewed the evolution of Indian automotive e-commerce and identified trust deficit, inspection uncertainty, and pricing opacity as the three primary barriers to digital transaction completion. Subsequent platforms addressing these barriers — Cars24's guaranteed buyback, CarDekho's certification programmes, Spinny's fixed-price model — achieved significant transaction volume growth but continued to prioritise transaction mechanics over user interface quality.

### *B. Progressive Web Applications*

Progressive Web Apps, introduced by Google engineers Russel and Berriman in 2015, combine the discoverability of web applications with the offline capability and performance of native mobile apps through service worker caching, Web

automotive lending banks (SBI, HDFC, ICICI, Axis, Kotak, Bank of Baroda, Punjab National Bank, Union Bank) via a cached proxy endpoint updated every 6 hours. The EMI computation implements the standard amortisation formula: EMI = P × r × (1+r)^n / ((1+r)^n − 1), where P is the principal (listing price minus down payment), r is the monthly interest rate, and n is the loan tenure in months. Users can adjust down payment percentage (10–50%), tenure (12–84 months), and preferred lender — the calculator dynamically recomputes EMI and displays a monthly payment breakdown with total interest paid [17].

### *E. Four-Car Comparison Tool*

The comparison tool enables side-by-side specification comparison of up to four vehicles simultaneously. Specification data for 28 attributes (engine, transmission, fuel efficiency, dimensions, safety features, entertainment system, warranty, service cost index) is sourced from a curated database of 6,200 Indian car model-year-variant combinations. Attribute rows are colour-coded to highlight the best value in each category — green for the winner, amber for middle values, and red for the worst value — using a configurable comparison direction (higher-is-better or lower-is-better per attribute). Cars are added to comparison from any listing card via a persistent bottom comparison tray that slides up from the bottom of the viewport, maintaining context without navigating away from the current listing page [18].

## V. DESIGN SYSTEM AND ACCESSIBILITY

### *A. Component Library*

AutonationIndia's UI is built on a custom design system comprising 48 reusable React components documented in Storybook 8.0. The design system implements a semantic colour token system mapping design intent (primary, secondary, success, warning, danger, neutral) to specific Tailwind CSS colour values, enabling consistent theming and dark mode support through CSS custom properties. Typography uses the Inter variable font for English and Poppins for Devanagari script, served as WOFF2 with font-display: swap to prevent render-blocking. The spacing system uses a 4px base grid, and the responsive breakpoint system defines four breakpoints: 375px (mobile), 768px (tablet), 1024px (desktop), 1440px (large desktop) [19].

### *B. Multilingual Support*

Internationalisation is implemented using react-i18next 14.0 with locale files for English (en), Hindi (hi), and Marathi (mr). Translation keys cover 840 UI strings including all button labels, filter labels, error messages, and static content. The language switcher in the navigation header persists the

App Manifest, and Push Notification APIs [5]. Biørn-Hansen et al. [6] conducted a systematic comparison of PWA, native, and hybrid app performance on Android devices and found that PWAs achieved within 15% of native app rendering performance on modern hardware while eliminating app store distribution friction. For automotive marketplaces where catalogue breadth and search experience are primary value drivers, PWA architecture enables offline catalogue browsing — a significant advantage in bandwidth-constrained Indian environments.

### *C. AI Price Estimation in Used-Car Markets*

Predictive pricing models for used vehicles have been studied extensively in the Western automotive context. Gegic et al. [7] achieved $R^2 = 0.91$ using gradient-boosted trees (XGBoost) on the UCI Automobile Dataset with features including mileage, age, fuel type, transmission, and brand. Pal et al. [8] applied random forest regression to Indian used-car data from CarDekho and achieved MAPE of 8.3% on a 6,019-listing dataset. The novelty of the AutonationIndia price estimator lies in its India-specific feature engineering — incorporating local RTO registration zone, CNG/LPG retrofitting status, Nashik vs. Mumbai registration premium, and seasonal demand adjustment factors absent from published models.

### *D. Core Web Vitals and Web Performance*

Google's Core Web Vitals framework [9], introduced in 2020 and incorporated into search ranking algorithms in 2021, established three user-centric performance metrics as industry standards: Largest Contentful Paint (LCP) measuring loading performance, Cumulative Layout Shift (CLS) measuring visual stability, and Interaction to Next Paint (INP) measuring interactivity. Pollard [10] demonstrated a strong correlation ($r = 0.74$) between LCP improvements and e-commerce conversion rate on mobile devices. The automotive marketplace context — characterised by image-heavy listing pages and complex filtering interactions — makes Core Web Vitals optimisation particularly challenging and commercially significant.

## III. SYSTEM ARCHITECTURE

### *A. Frontend Technology Stack*

AutonationIndia is built on React.js 18.3 with TypeScript 5.4, utilising the new concurrent rendering capabilities (useTransition, useDeferredValue) introduced in React 18 to ensure filter interactions remain responsive during heavy computation. Vite 5.2 serves as the build tool, replacing Create React App for its significantly faster hot module replacement and optimised production bundling. Tailwind CSS 3.4 provides selected locale in localStorage and applies it on subsequent sessions. Marathi locale uses Devanagari numerals (० १ २ ३) for price display and formats currency in lakh/crore notation (e.g., ₹8.5 लाख) rather than Western thousand-separator format. The HTML lang attribute and dir attribute are dynamically updated on locale change for correct screen reader behaviour [20].

### *C. WCAG 2.1 Level AA Compliance*

Accessibility testing used axe-core 4.9 automated scanning supplemented by manual testing with NVDA screen reader on Windows and VoiceOver on iOS. All interactive elements maintain a minimum touch target size of 44×44 pixels per WCAG 2.5.5. Colour contrast ratios exceed 4.5:1 for normal text and 3:1 for large text and UI components. All images include descriptive alt text generated automatically from listing data and editable by sellers. Form inputs are associated with visible labels and include descriptive aria-describedby attributes for validation error messages. Keyboard navigation is fully supported with visible focus indicators achieving 3:1 contrast with adjacent colours [21].

## VI. PERFORMANCE EVALUATION

### *A. Lighthouse Audit Results*

Table II presents comprehensive Lighthouse audit results for AutonationIndia's primary pages (Home, Listing Search, Car Detail, Comparison) averaged across three audit runs on a Moto G Power device profile (4x CPU slowdown, throttled 4G connection) using Chrome DevTools Lighthouse 11.0. All four Lighthouse category scores exceed 90, with perfect SEO score of 100 attributable to structured data markup (Schema.org Vehicle and Offer schemas), complete meta tag implementation, and XML sitemap generation. The Performance score of 94 is achieved through aggressive code splitting (16 lazy-loaded route chunks), image optimisation (WebP with AVIF fallback, srcset for responsive images), and critical CSS inlining [22].

### *B. Core Web Vitals Analysis*

FCP of 1.2 s is achieved by inlining the above-the-fold hero component's CSS, preloading the hero image, and eliminating render-blocking resources through async script loading. LCP of 2.1 s on the listing search page is achieved by prioritising the first visible listing card image using loading='eager' and fetchpriority='high' attributes. TBT of 48 ms — significantly below the 200 ms threshold — results from React 18's concurrent rendering splitting long tasks into smaller chunks using time-slicing. CLS of 0.04 is achieved by specifying explicit width and height attributes on all images (preventing layout reflow on load) and reserving skeleton

the utility-first styling system, enabling consistent spacing, typography, and colour tokens across the component library without runtime CSS-in-JS overhead. React Router 6 handles client-side navigation with data-loading route loaders that prefetch listing data before route transition completes [11].

### *B. State Management with Zustand*

Global state management uses Zustand 4.5 — a lightweight, hooks-based state library chosen over Redux Toolkit for its dramatically lower boilerplate while maintaining predictable unidirectional data flow. Three Zustand stores manage distinct domains: FilterStore (15 filter parameters, active filter count, filter reset), CarListingStore (paginated listing results, loading states, sort order), and ComparisonStore (selected cars for comparison, maximum 4 cars, comparison modal state). The filter store's subscribe() mechanism enables URL synchronisation of filter state, ensuring shareable filtered search URLs — a missing feature in CarDekho and OLX Autos that prevents users from sharing specific search results [12].

### *C. PWA Implementation*

PWA capability is implemented using Workbox 7.0 via the vite-plugin-pwa integration. The service worker implements a hybrid caching strategy: Cache First for static assets (JS bundles, CSS, fonts, icons) with a 30-day TTL; Network First with cache fallback for listing API responses with a 5-minute TTL; and Stale While Revalidate for car images with background refresh. The Web App Manifest specifies standalone display mode, theme colour (#1a56db — the AutonationIndia brand blue), and adaptive icons for both light and dark mode. Offline mode displays cached listings with a visual indicator distinguishing live from cached results, enabling users to continue browsing in areas with intermittent connectivity [13].

## IV. KEY FEATURE IMPLEMENTATION

### *A. 15-Parameter Advanced Filtering Engine*

The filtering engine supports simultaneous filtering across 15 parameters organised into four categories: Vehicle Identity (make, model, variant, body type, colour); Condition (year range, mileage range, number of owners, accident history); Financials (price range, fuel type, insurance validity); and Location (city, registration zone, home delivery available). Filter state is encoded in the URL query string using a custom serialiser that compresses multi-value parameters. Filter application triggers an optimistic UI update using React's useTransition hook — the filter panel updates immediately while the listing grid re-renders asynchronously, eliminating the perceived latency of filter operations. Filter combinations screen space for dynamically loaded content [23].

### *C. Bundle Analysis*

The production JavaScript bundle is 142 KB gzipped (486 KB uncompressed), analysed using rollup-plugin-visualizer. The React and ReactDOM runtime accounts for 42 KB (29.6%), Three.js for 38 KB (26.7%), Zustand and React Router for 8 KB combined, and application code for the remaining 54 KB. Three.js is loaded only when the 360° viewer is opened using React.lazy() and Suspense, preventing its 38 KB cost from impacting initial page load. Chart.js for the EMI payment visualisation is similarly lazy-loaded on demand [24].

## VII. USER TESTING AND EVALUATION

User testing was conducted with 42 participants recruited from Dr. Moonje Institute's student and faculty community and local Nashik residents who had expressed intent to purchase a used vehicle within 6 months. Participant demographics: 26 male, 16 female; age range 21–52 years; 31 using smartphones as primary internet device; 18 preferring Marathi as primary language. Testing sessions were conducted in-person on participants' own devices using the deployed AutonationIndia prototype, following a think-aloud protocol with 6 defined tasks. Sessions were screen-recorded with participant consent for post-session analysis. The System Usability Scale (SUS) questionnaire [25] was administered after task completion, yielding a mean score of 86.4 — classified as 'Excellent' on the Sauro-Lewis adjective rating scale (scores > 85.5). Table III presents task-level success rates and completion times. The lowest task success rate (88.1%) was observed for the 360° viewer feature, attributable to discoverability — 5 participants did not notice the 360° button on the listing detail page. Post-test interviews revealed that Marathi language support was the most appreciated feature among regional participants, with 14 of 18 Marathi-preferring participants explicitly commenting on its importance for understanding technical specifications [26].

## VIII. CONCLUSIONS AND FUTURE WORK

This paper has presented AutonationIndia, a high-performance, mobile-first, accessible, and multilingual frontend for an Indian used-car marketplace, demonstrating that modern web technologies can deliver a user experience substantially superior to incumbent platforms on all measurable dimensions. Lighthouse scores of 94/98/96/100 across Performance/Accessibility/Best Practices/SEO, Core Web Vitals compliance on all four metrics, and a SUS score of 86.4 establish AutonationIndia as a reference

with zero results display intelligent suggestions for relaxing individual parameters rather than an empty state [14].

### *B. AI Price Estimation Module*

The price estimation backend uses a Gradient Boosted Regression Tree model (XGBoost 2.0) trained on 184,000 Indian used-car transaction records collected from publicly available CarDekho listings between January 2022 and December 2024. Feature engineering incorporates 23 input variables: make, model, variant, manufacturing year, registration year, fuel type, transmission, mileage, number of owners, RTO city, insurance type, colour, tyre condition, and six India-specific derived features (monsoon penalty, CNG kit status, Nashik vs. Mumbai price premium, festive season demand index). The model achieves MAPE of 6.8% on a held-out test set of 18,400 listings — superior to the published 8.3% of Pal et al. [8]. The model is served via a FastAPI endpoint deployed on Render.com; the frontend displays the estimated price range as a confidence interval with a visual gauge indicating whether the listed price is fair, below market, or above market [15].

implementation for high-quality automotive marketplace frontend engineering in the Indian context. The AI price estimation module's MAPE of 6.8% on 18,400 Indian car listings, and the Three.js 360° inspector, provide differentiated features unavailable on any existing Indian used-car platform.

Future work will address four areas: (1) integration of a large language model (LLM) powered conversational search interface enabling natural language queries such as 'family car under 6 lakhs with low fuel cost in Nashik' processed using a fine-tuned Gemma-2B model; (2) AR try-on feature using WebXR API enabling users to visualise how a selected car looks in their home driveway using their smartphone camera; (3) seller verification dashboard integrating DigiLocker API for RC document verification and Aadhaar-based seller identity authentication; (4) expansion of the regional multilingual support to Gujarati, Tamil, and Kannada to support deployment in additional high-growth Tier-2 markets.

**TABLE I — Feature and Performance Comparison: AutonationIndia vs. Major Indian Used-Car Marketplace Platforms**

| Feature | AutonationIndia | CarDekho | Cars24 | OLX Autos | CarWale |
|---|---|---|---|---|---|
| AI Price Estimator | ✓ | ✓ | ✓ | ✗ | ✓ |
| Advanced Filter (15+ params) | ✓ | ✗ | ✗ | ✗ | Partial |
| 360° Virtual Car View | ✓ | ✗ | ✗ | ✗ | ✗ |
| EMI Calculator (live rates) | ✓ | ✓ | ✓ | ✗ | ✓ |
| Compare Side-by-Side (4 cars) | ✓ | ✗ | ✓ | ✗ | ✓ |
| Responsive (Mobile-First) | ✓ | Partial | ✓ | ✓ | Partial |
| Offline Mode (PWA) | ✓ | ✗ | ✗ | ✗ | ✗ |
| Marathi / Hindi UI | ✓ | Hindi only | ✗ | ✗ | Hindi only |
| Lighthouse Score | 94 | 71 | 78 | 68 | 74 |
| First Contentful Paint | 1.2 s | 3.1 s | 2.4 s | 3.8 s | 2.9 s |

**TABLE II — Google Lighthouse Audit and Core Web Vitals Results — AutonationIndia (Moto G Power Profile, Throttled 4G)**

| Metric | Score / Value | Target / Threshold | Status |
|---|---|---|---|
| Lighthouse Performance Score | 94 / 100 | ≥ 90 | ✓ PASS |
| Lighthouse Accessibility Score | 98 / 100 | ≥ 90 | ✓ PASS |
| Lighthouse Best Practices | 96 / 100 | ≥ 90 | ✓ PASS |
| Lighthouse SEO Score | 100 / 100 | ≥ 90 | ✓ PASS |
| First Contentful Paint (FCP) | 1.2 s | ≤ 1.8 s | ✓ PASS |
| Largest Contentful Paint (LCP) | 2.1 s | ≤ 2.5 s | ✓ PASS |

| | | | |
|---|---|---|---|
| Total Blocking Time (TBT) | 48 ms | ≤ 200 ms | ✓ PASS |
| Cumulative Layout Shift (CLS) | 0.04 | ≤ 0.10 | ✓ PASS |
| Time to Interactive (TTI) | 2.8 s | ≤ 3.8 s | ✓ PASS |
| Bundle Size (gzip) | 142 KB | ≤ 200 KB | ✓ PASS |
| WCAG 2.1 Compliance | AA | AA | ✓ PASS |

**TABLE III — User Testing Results: Task Success Rates, Completion Times, and System Usability Scale Score (n = 42)**

| Task / Usability Metric | Participants | Success Rate (%) | Avg. Time (seconds) | SUS Score (/ 100) |
|---|---|---|---|---|
| Search and filter a specific car | 42 | 97.6 | 18.4 | — |
| Compare two cars side-by-side | 42 | 92.8 | 24.6 | — |
| Calculate EMI for a selected car | 42 | 100.0 | 12.2 | — |
| Locate seller contact / book test drive | 42 | 95.2 | 21.8 | — |
| Use 360° view feature | 42 | 88.1 | 31.4 | — |
| Navigate on mobile (375 px viewport) | 42 | 100.0 | — | — |
| Overall System Usability Scale (SUS) | 42 | — | — | 86.4 |

## IX. REFERENCES

[1] KPMG India, India's Used Car Market: The Road to 2030, KPMG Advisory Services, Mumbai, 2024.

[2] Google PageSpeed Insights, Performance Audit Reports for CarDekho, Cars24, OLX Autos, CarWale, Spinny, Retrieved March 2025.

[3] Society of Indian Automobile Manufacturers, Annual Report on Vehicle Registration Statistics, SIAM, New Delhi, 2024.

[4] Subramanian K. and Hariharan P., Digital transformation of used vehicle markets in India, Journal of Business Research, vol. 142, pp. 1022–1036, 2022.

[5] Berriman R. and Russell A., Progressive Web Apps: Escaping Tabs Without Losing Our Soul, Google Developers Blog, 2015.

[6] Biørn-Hansen A. et al., Progressive web apps: The possible web-native unifier for mobile development, Proc. WEBIST, pp. 344–351, 2017.

[7] Gegic E. et al., Car price prediction using machine learning techniques, TEM Journal, vol. 8, no. 1, pp. 113–118, 2019.

[8] Pal N. et al., Used car price prediction using machine learning with the CarDekho dataset, Procedia Computer Science, vol. 218, pp. 1208–1218, 2023.

[9] Google Developers, Core Web Vitals — Web.dev Documentation, Google LLC, Retrieved April 2025.

[10] Pollard A., The Business Impact of Core Web Vitals: A Correlation Study, Deloitte Digital Insights, 2021.

[11] Facebook Inc., React Documentation v18 — Concurrent Features, React.dev, 2024.

[12] Zustand GitHub Repository, pmndrs/zustand, v4.5.0, GitHub Inc., 2024.

[13] Google Chrome Team, Workbox Documentation v7.0, Google LLC, 2024.

[14] Wieringa R., Design Science Methodology for Information Systems and Software Engineering, Springer, Berlin, 2014.

[15] Chen T. and Guestrin C., XGBoost: A scalable tree boosting system, Proc. ACM KDD, pp. 785–794, 2016.

[16] Three.js Documentation, Three.js r158 — WebGL Javascript 3D Library, threejs.org, 2024.

[17] Reserve Bank of India, DBIE — Database on Indian Economy, Public Data API, rbi.org.in, 2025.

[18] Nielsen J., Usability Engineering, Morgan Kaufmann, San Francisco, CA, 1994.

[19] Storybook Documentation v8.0, Component Driven Development, storybook.js.org, 2024.

[20] i18next, react-i18next v14.0 Documentation, i18next.com, 2024.

[21] W3C Web Accessibility Initiative, Web Content Accessibility Guidelines (WCAG) 2.1, W3C Recommendation, 2018.

[22] Google Developers, Lighthouse 11.0 Documentation, Google LLC, 2024.

[23] Walton P., Defining the Core Web Vitals metrics thresholds, Web.dev, Google LLC, 2020.

[24] Rollup Plugin Visualizer, btd/rollup-plugin-visualizer v5.12, GitHub Inc., 2024.

[25] Brooke J., SUS — A quick and dirty usability scale, in Usability Evaluation in Industry, Taylor and Francis, pp. 189–194, 1996.

[26] Sauro J. and Lewis J.R., Quantifying the User Experience: Practical Statistics for User Research, 2nd ed., Morgan Kaufmann, 2016.

[27] Google Developers, WebXR Device API Specification, W3C Working Draft, 2024.

[28] Ministry of Electronics and Information Technology, DigiLocker API Documentation, Government of India, 2024.

[29] Gulave A. and Darne H., AutonationIndia: Source code and design system, GitHub Repository, github.com/autonationindia, 2025.

[30] Google, Schema.org Vehicle Markup for Automotive SEO, Schema.org/Vehicle, 2024.


ISSN: Applied For (2026) | DOI: 10.XXXXX/IJVAST.2026.XXXXX | Open Access — CC BY 4.0 License