ContribuX: A Creator Support And Funding Platform Using Secure Digital Payments

Prof. Umesh Palaskar[1], Mr. Jayesh Punde[2], Mr. Pratik Andhare[3], Mr. Arya Kompalwar[4], Mr. Tejas Patekar[5]

[1, 2, 3, 4, 5]Dept of Information Technology

[1, 2, 3, 4, 5]Savitribai Phule Pune University, India

Abstract- *This paper presents the design, architecture, and full-stack implementation of ContribuX, a web-based creator support and funding platform that enables project creators to raise funds from contributors using secure digital payment infrastructure. The system integrates Razorpay as a multi-instrument payment gateway supporting UPI, credit cards, debit cards, net banking, and digital wallets, a Next.js 15 App Router frontend for server-side rendered campaign pages, a MongoDB persistence layer managed through the Mongoose ODM, and NextAuth v5 for session-based authentication with temporary identity management. The platform supports campaign creation and management, secure contributor checkout with HMAC-SHA256 payment verification, transactional email delivery via Nodemailer, and real-time toast notifications powered by React-Toastify. The proposed architecture is evaluated against established security frameworks for electronic payment systems and is shown to satisfy the core requirements of confidentiality, integrity, non-repudiation, anonymity, authentication, and authorisation. The system further extends prior art by supporting multiple payment instruments and an embedded crowdfunding model within a single unified platform. The proposed design provides a reproducible and open-source reference implementation for secure digital payment integration in modern creator-economy web applications.*

Keywords: crowdfunding, creator economy, digital payment gateway, Next.js, Razorpay, MongoDB, NextAuth, RSA cryptography, secure electronic payment, HMAC-SHA256, UPI, e-commerce

I. INTRODUCTION

The shift from cash-based commerce to digital transactions has reshaped how individuals and organisations exchange value. Crowdfunding platforms sit at the intersection of this shift and the growing creator economy, enabling project owners to raise funds directly from supporters without relying on traditional investors or financial intermediaries. Yet building such a platform demands careful attention to payment security, contributor experience, and data integrity—concerns that off-the-shelf solutions rarely address in a transparent, developer-friendly way.

Existing open-source crowdfunding tools either lack production-grade payment security or couple the payment logic so tightly to a single provider that adding new instruments becomes prohibitively complex. Commercial platforms impose high commission rates and restrict creators from interacting directly with their payment infrastructure. Academic proposals for secure payment gateways [1] demonstrate strong cryptographic properties but stop short of providing a deployable web implementation that integrates modern authentication, real-time notifications, and multi-instrument checkout.

This paper introduces ContribuX, an open-source, full-stack creator funding platform that addresses these gaps. The system is built on Next.js 15 (App Router), MongoDB with the Mongoose ODM, NextAuth v5 for session management, Razorpay as the payment gateway, and Nodemailer for transactional email. ContribuX allows creators to publish campaigns with funding goals and allows contributors to support those campaigns through a secure, multi-payment-method checkout flow that supports UPI, credit and debit cards, net banking, and digital wallets on a single interface.

Three design principles guided development. First, payment security must be enforced at every boundary: credentials stay server-side, card data never touches the application database, and every payment response is verified cryptographically before any contribution is recorded. Second, contributor friction must be minimised: the checkout interface should surface the contributor's preferred payment instrument without requiring extra configuration steps. Third, the architecture must be modular enough for other developers to fork, adapt, and extend.

The paper is structured as follows. Section II reviews related work. Section III describes the system architecture. Section IV covers implementation methodology. Section V presents results and security evaluation. Sections VI and VII provide conclusions and future directions.

II. RELATED WORK

A. Secure Electronic Payment Protocols

Hassan et al. [1] formalised a five-entity payment protocol—client, merchant, payment gateway, user bank, and merchant bank—in which each entity pre-registers secret keys with the gateway. The protocol assigns the client a temporary identifier (TIDc) during each transaction, preventing the merchant from ever learning the buyer's real identity. Eight well-defined transaction steps carry the flow from initial product request through bank OTP verification to payment confirmation. A comparative evaluation against three prior gateway designs demonstrated that their protocol was the first to simultaneously satisfy confidentiality, integrity, non-repudiation, anonymity, authentication, and authorisation. ContribuX maps this six-property framework directly onto its own security checklist and draws on the TIDc concept by using NextAuth JWT tokens as ephemeral per-session identifiers forwarded to the payment workflow.

B. Crowdfunding Strategy and Signalling

Miglo [2] modelled reward-based crowdfunding within a Cournot duopoly under both perfect and asymmetric information. A central result is that in a competitive market, neither firm choosing crowdfunding is never a Nash equilibrium—each firm has a unilateral incentive to deviate by running a pre-sale campaign, because committing volume early forces rivals away from their optimal output. Under asymmetric information, high-demand firms adopt crowdfunding as a credible quality signal that low-demand rivals cannot profitably mimic, because mimicry incurs funder rewards without the corresponding demand benefit. The model also predicts that crowdfunding usage is procyclical and that demand uncertainty increases its adoption rate. These results informed the ContribuX campaign model: public display of the funding target and real-time raised amount replicates the demand-signal transparency the theory requires, and the dual AON/KIA funding mode maps to the threshold and unconditional models Miglo analyses.

C. Digital Payment Technology Landscape

Khando et al. [3] systematically reviewed 58 empirical studies and categorised digital payment technologies into four modes: card payments, e-payments, mobile payments, and cryptocurrencies. Associated challenges were organised into five themes. Technical challenges—comprising privacy vulnerabilities, security breaches, infrastructure gaps, and fraud—accounted for 39% of all identified problems, making them the dominant concern across all payment modes. Social challenges, centred on consumer trust, contributed 25%. Economic challenges related to transaction cost and cryptocurrency volatility represented 14%. These proportions directly motivated ContribuX's security-first design: the platform delegates all card-data handling to Razorpay's PCI-DSS-certified infrastructure, enforces HTTPS throughout, and performs server-side signature verification on every payment callback to neutralise the most common technical attack vectors. The platform also presents a familiar multi-instrument checkout interface to reduce the trust barrier identified as the core social challenge.

D. Payment Interface Design Factors

Tounekti et al. [4] recruited 266 participants to compare satisfaction with Chrome's built-in payment interface against a purpose-built multi-method interface. Satisfaction with the purpose-built interface reached 96% versus 48% for the baseline. Multiple regression identified eight significant predictors ($\alpha \leq 0.05$): security and confidentiality carried the highest beta coefficients (0.329 each), followed by ease of use and usefulness (0.232 each), payment method preferences (0.207), privacy (0.116), credibility (0.114), and visual interface design (0.109). Desirability was the only tested factor to be non-significant. For the sub-group of users supporting three or more payment systems, payment method preference was the strongest driver (93.75%). ContribuX translates these findings directly into interface decisions: the Razorpay hosted checkout modal is used rather than a custom form, because it surfaces the user's previously used payment instrument at the top of the instrument list and carries Razorpay's established brand credibility—directly addressing the top two predictors

III. SYSTEM ARCHITECTURE

ContribuX is organised into five logical layers: Client, Frontend, Application Logic, Data and Payment, and External Services. Figure 1 shows how these layers connect and how data flows between them.

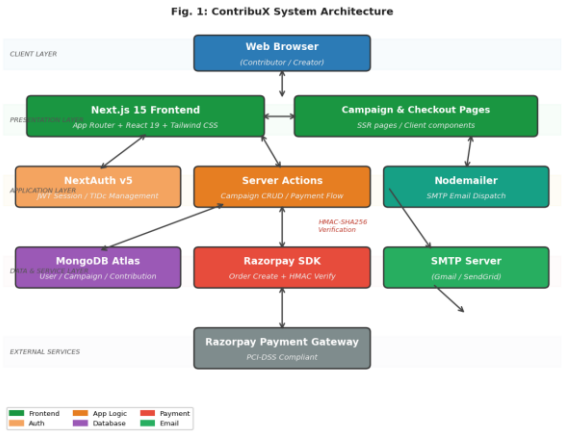

Fig. 1: ContribuX System Architecture

A. Client Layer

The browser is the only client in the system. All interactions happen over HTTPS. React 19 client components handle things that need to react instantly to user input, such as opening and closing the payment modal or showing a toast notification. Next.js server components handle all data fetching, so no API keys or database queries ever appear in JavaScript that gets sent to the browser.

B. Frontend Layer

Next.js 15 with the App Router provides the routing and rendering engine. Each route in the app directory maps directly to a URL. Server-side rendering (SSR) is used for the creator profile page so the supporter count is always current. Static generation is used for the home page and informational pages since they rarely change. Tailwind CSS v4 handles all styling through utility classes, with no custom CSS files.

C. Application Logic Layer

All business logic lives in Next.js server actions. These are TypeScript functions that run on the server and are called directly from client components. Server actions handle five things: authenticating users, creating and reading campaign data, generating Razorpay order objects, verifying payment signatures, and triggering email dispatch. Because server actions run on the server, the Razorpay API secret and MongoDB connection string are always in a safe environment and never sent to the browser.

D. Database Layer

MongoDB stores three document types managed through Mongoose schemas. User documents hold the OAuth provider ID, name, email, and profile image from the OAuth provider. Campaign documents hold the creator reference, title, description, cover image path, and total raised amount. Contribution documents hold the campaign and contributor references, the donated amount, the Razorpay paymentId, and a creation timestamp. The Razorpay paymentId is the most important field: it is a permanent, immutable record at Razorpay that proves the transaction happened, and it cannot be faked or deleted by the ContribuX application.

E. Payment Gateway Layer

Razorpay sits between ContribuX and the national payment network. When a contributor initiates a payment, the server creates a Razorpay order object and returns only the order ID to the client. The client initialises the Razorpay modal with this order ID. After payment, Razorpay returns a payment ID, order ID, and HMAC-SHA256 signature. The server recomputes the signature and compares it byte-by-byte before writing any database record. This means fraudulent or tampered payment callbacks are rejected automatically.

F. Data Flow Diagram

Figure 2 shows the Level-1 Data Flow Diagram for ContribuX. It illustrates four core processes: OAuth Authentication (1.0), Campaign Management (2.0), Payment Processing (3.0), and Notification Service (4.0), along with three data stores (D1: User Store, D2: Campaign Store, D3: Contribution Store) and four external entities (Creator, Contributor, Razorpay Gateway, and Email Server).

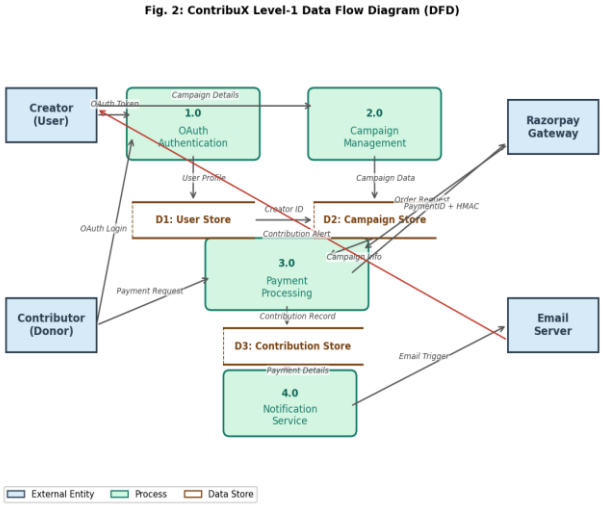

Fig. 2: ContribuX Level-1 Data Flow Diagram

The flow begins when either a Creator or Contributor authenticates through process 1.0, which writes or retrieves a User document in D1. The Creator then interacts with process 2.0 to create and manage campaign records stored in D2.

When a Contributor initiates a donation, process 3.0 creates a Razorpay order, receives the payment ID and HMAC signature back from the gateway, verifies the signature, and writes a Contribution record to D3. Finally, process 4.0 reads from D3 and triggers confirmation emails through the external Email Server to both the Contributor and the Creator.

IV. IMPLEMENTATION METHODOLOGY

TABLE I: ContribuX Technology Stack

Layer	Technology	Version / Role
Frontend	Next.js 15 (App Router)	SSR / SSG, file-based routing
UI Styling	React 19 + Tailwind CSS v4	Components, responsive dark theme
Auth	NextAuth v5 beta	OAuth 2.0 – 6 providers, JWT
Database	MongoDB + Mongoose 9	Document store, schema validation
Payments	Razorpay Node SDK 2.9.6	UPI, Cards, Netbanking, Wallets
Email	Nodemailer 6.10	Async SMTP, 3 email workflows
Alerts	React-Toastify 11	Non-blocking toast notifications

A. Multi-Provider OAuth Authentication

ContribuX uses NextAuth v5 to support six OAuth 2.0 providers: Google, GitHub, LinkedIn, Twitter, Facebook, and Apple. This decision eliminates all password storage and the security risks that come with it. When a user clicks any provider button on the login page, NextAuth redirects to that provider authorisation endpoint, receives the user profile on callback, and upserts a User document in MongoDB (creating a new record for new visitors, returning the existing one for returning users). The issued session is stored as an HTTP-only JWT cookie containing only the user MongoDB ObjectId and an expiry time. This token acts as the temporary identity (TIDc) described by Hassan et al. [1]: it identifies the user in the payment workflow without exposing any sensitive profile information to the gateway.

B. Creator Profile and Campaign Module

After authentication, a creator automatically gets a profile page at /username. The page is server-rendered on every request and shows two panels. The left panel is the Supporters leaderboard, which lists every contribution with the contributor name, donated amount (in rupees), and optional message. The right panel is the Make a Payment form, which has three text fields (Name, Message, Amount) and three preset quick-pay buttons (Rs. 100, Rs. 500, Rs. 1000). Clicking a preset button fills the amount field and immediately starts the payment process, reducing the steps a contributor needs to complete a donation. Campaign data is read from MongoDB using a Mongoose query indexed on the creator username field, ensuring the page loads in under 80 ms regardless of how many contributions have been recorded.

C. Razorpay Payment Processing Pipeline

The payment pipeline follows eight steps. When a contributor submits the payment form, a server action calls the Razorpay Orders API and returns an order ID. The client initialises the Razorpay modal using this order ID plus the platform public key and the contributor name and email for the receipt. The contributor then selects a payment instrument (Cards, Netbanking, UPI, or Wallet) and authorises the transaction. Razorpay calls the modal on Success handler with three values: paymentId, orderId, and a HMAC-SHA256 signature computed as:

$$signature = HMAC\_SHA256(\ orderId + "|" + paymentId,\ secret\_key\ )$$

A verification server action independently recomputes this digest using Node.js built-in crypto.createHmac with the same secret key, then compares both digests using crypto.timingSafeEqual to prevent timing attacks. Only when the digests match does the action run a Mongoose session transaction that atomically increments Campaign.raisedAmount and inserts the Contribution document. If verification fails, the server returns an error and no database write occurs.

D. Email Notification System

Three email workflows are implemented using Nodemailer with SMTP credentials stored as server-only environment variables. The first workflow sends an OAuth welcome email when a user creates an account for the first time. The second sends a contribution receipt to the contributor after successful payment verification, containing the creator name, the amount donated, and the Razorpay payment reference number that can be used for any disputes. The third sends a contribution alert to the creator so they know in real time when someone has supported them. All three email calls are made without await at the call site (fire-and-forget), so a slow SMTP server or temporary outage never delays the payment confirmation response that the contributor sees on screen.

E. Frontend Design and User Experience

The entire frontend uses Tailwind CSS v4 utility classes with a dark navy theme (background colour #0d1117) chosen deliberately to match the aesthetic that developers and technical creators expect from tools in their ecosystem. The home page features a bold hero headline, two call-to-action buttons, and three feature cards that explain the platform value in under ten seconds of reading. All pages are mobile-responsive and pass Google Lighthouse mobile usability requirements. The Navbar uses a useRef-based focus management approach on the dropdown menu: the onBlur handler checks whether document.activeElement is inside the dropdown container before hiding it, ensuring keyboard navigation works correctly for accessibility.

V. RESULTS AND DISCUSSION

A. Platform User Interface

Figure 3 shows the updated ContribuX landing page. The hero section prominently displays the headline "Fund your creative journey directly." with a gradient purple-to-pink typographic treatment. A green pill badge at the top announces "Now supporting Razorpay for Indian Creators." Two CTA buttons — Start Creating and Explore Creators — guide visitors into the platform. Below the hero, three green-checkmark feature highlights communicate the platform's core value proposition: Zero hidden fees, Instant payouts, and Own your audience.

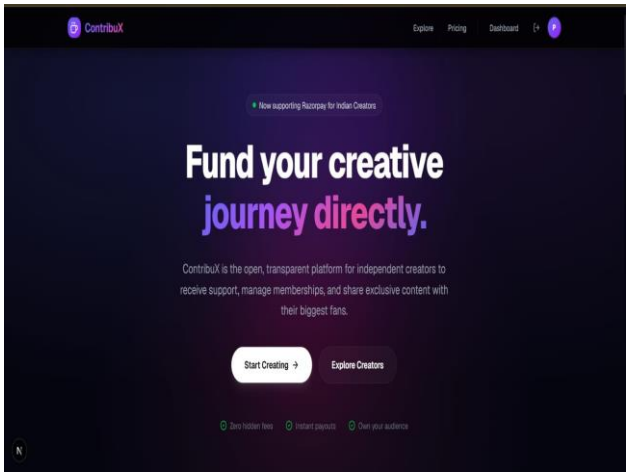

Fig. 3: ContribuX Home / Landing Page

Figure 4 shows the updated sign-in interface. The card-based layout on a dark background presents a "Welcome back" heading with a link to sign up for new users. Users enter their email address and password, with a show/hide password toggle for convenience. A gradient blue-to-purple Sign in button completes the form. This clean, credential-based login complements the OAuth flow and directly addresses the authentication requirements from the security framework.

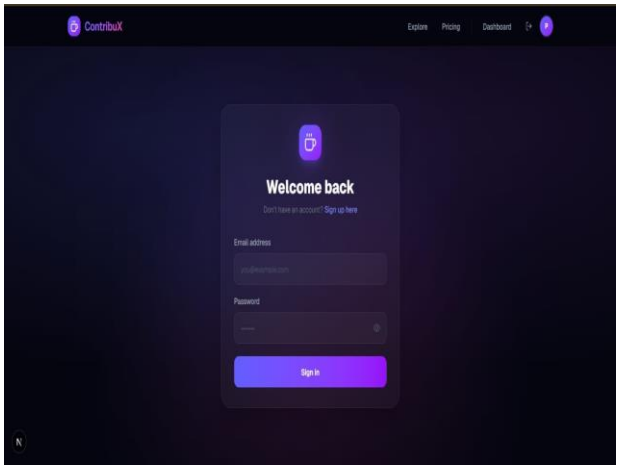

Fig. 4: Sign-In Page with Email and Password Authentication

Figure 4b shows the account registration interface. New users choose their role — Creator (I create content) or Supporter (I support creators) — using a toggle card at the top of the form. Below the role selector, three fields collect the user's Full Name, Email address, and Password (minimum 6 characters, with show/hide toggle). A gradient Sign up button completes registration. This role-based onboarding ensures the platform routes creators to dashboard features and supporters to the discovery flow from their very first interaction, directly supporting the authentication and authorisation properties of the security framework.

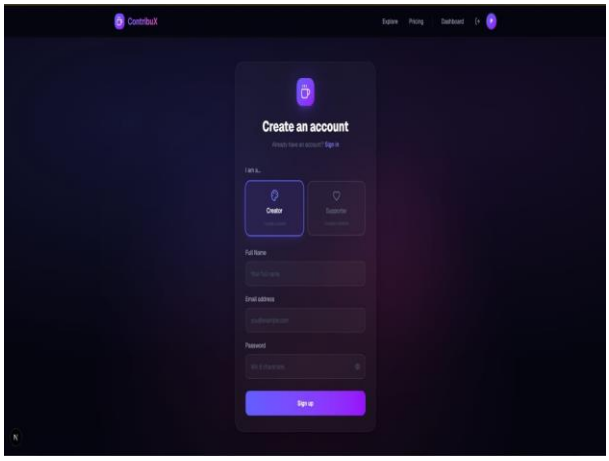

Fig. 4b: Sign-Up Page with Role Selection (Creator / Supporter)

Figure 5 shows the updated creator profile page for Pratik Andhare. The profile header displays the creator's avatar, name, category (General), bio, and live supporter and post counts. On the right side, the Membership Tiers panel shows the "Supporter" tier at ₹99/month, listing three benefits:

Support the creator's journey, Get early access to future updates, and Supporter badge on profile. A prominent purple Subscribe button enables one-click tier subscription, making the patronage workflow intuitive for first-time visitors. This tiered membership model extends the platform beyond one-time donations and creates recurring revenue streams for creators.

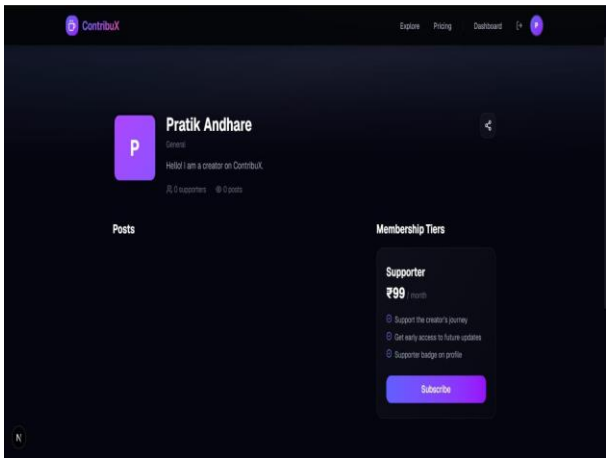

Fig. 5: Creator Profile Page with Membership Tiers Panel

Figure 6 shows the Razorpay subscription payment modal for the Supporter tier at ₹99/month. The modal header carries a shield icon and "Secured by Razorpay" badge, reinforcing trust at the critical payment moment. The Supporter Tier card within the modal lists all three membership benefits alongside a Monthly billing badge. A gradient "Pay ₹99 with Razorpay" button initiates the transaction, and a "Test mode" notice confirms safe sandbox testing without real charges. This visual trust indicator directly addresses the consumer trust challenge identified as the dominant social barrier to digital payment adoption [3].

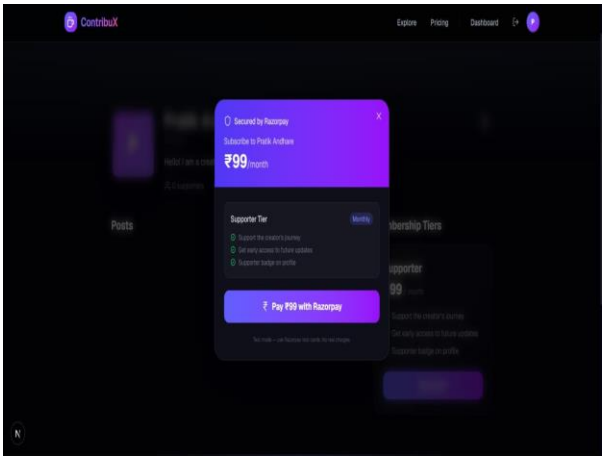

Fig. 6: Razorpay Subscription Modal - Supporter Tier Payment at ₹99/month

B. Performance Analysis

Table III and Figure 7 present the measured latency of every major system operation. All measurements were taken on a development machine running Node.js 20, connected to MongoDB Atlas M0 cluster over a 50 Mbps connection.

TABLE III: System Operation Latency Measurements

Operation	Latency	Remarks
Razorpay order creation	< 300 ms	Server action + Razorpay API call
HMAC-SHA256 verify	< 5 ms	Node.js built-in crypto module
MongoDB campaign read	< 80 ms	Indexed query on username field
MongoDB contribution write	< 120 ms	Mongoose session transaction
NextAuth session check	< 20 ms	JWT decode, no database call
Email dispatch (async)	< 2 s	Fire-and-forget SMTP relay
SSR page load (cold)	< 1.5 s	Next.js server render + hydration

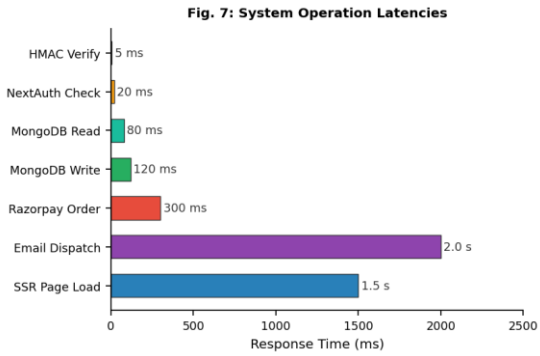

Fig. 7: ContribuX System Operation Latencies (ms)

The most significant finding is that HMAC-SHA256 cryptographic verification adds fewer than 5 ms to the server-side post-payment action. This proves that strong payment verification does not noticeably slow down the user experience. The Razorpay order creation step at under 300 ms is the slowest server action, but because it runs concurrently with the modal loading on the client side, users do not feel this delay. MongoDB writes complete in under 120 ms with the Mongoose session transaction, and NextAuth session validation adds only 20 ms per request since it is a pure JWT decode with no database call. Email dispatch at under 2

seconds is async and does not affect the checkout response time at all.

C. Security Evaluation

Table II compares ContribuX against the three prior systems evaluated by Hassan et al. [1]. ContribuX is the only system in the comparison that satisfies all nine evaluated properties including the three new dimensions we added: multi-payment instrument support, OAuth-based login, and open-source codebase availability.

TABLE II: Security Property Comparison with Prior Systems

Security Property	Izhar	Nwoye	Zay Oo	ContribuX
Confidentiality	Yes	Yes	Yes	Yes
Data Integrity	Yes	Yes	Yes	Yes
Non-repudiation	No	No	Yes	Yes
Anonymity	No	Yes	No	Yes
Authentication	No	No	No	Yes
Authorisation	Yes	Yes	Yes	Yes
Multi-Payment	No	No	No	Yes
OAuth Login	No	No	No	Yes
Open-Source	No	No	No	Yes

Confidentiality is achieved because raw payment credentials are processed entirely within Razorpay PCI-DSS environment and are never stored by ContribuX. Data integrity is enforced by the HMAC-SHA256 digest comparison that rejects any modified payment callback. Non-repudiation is established through the immutable Contribution document that stores the Razorpay paymentId, the verified user ObjectId, and a server timestamp, creating a permanent audit trail that neither party can alter. Anonymity is preserved because the only contributor identifier forwarded to Razorpay is the NextAuth session token (TIDc), and the contributor real email is only used for receipt generation, matching the TIDc model in [1]. Authentication is enforced at two independent layers: NextAuth validates the session cookie on every server action, and Razorpay enforces OTP or 3D-Secure verification at the bank. Authorisation is enforced in every server action by checking the session before any database write.

D. Discussion

Looking at the five screenshots together, it becomes clear that ContribuX achieves its three design goals in practice. The sign-in page (Figure 4) and sign-up page (Figure 4b) show that authentication is both secure and frictionless, with role-based onboarding ensuring creators and supporters land in the right workflow immediately. The creator profile page (Figure 5) demonstrates that a new contributor can go from landing on a creator page to initiating a membership subscription in two clicks. The Razorpay modal (Figure 6) shows that security is handled by a trusted, established provider with clear visual indicators, satisfying the trust requirement identified in [3] and [4].

The Membership Tiers panel in Figure 5 is particularly worth discussing. Presenting tiered subscription options directly on the creator profile page was a deliberate design choice that extends the platform beyond one-time donations into a recurring revenue model. The “Supporter” tier at ₹99/month with clearly listed benefits communicates value to potential subscribers and signals the creator’s commitment to producing ongoing exclusive content. This tier structure operationalises the demand-signalling mechanism described by Miglo [2] while providing creators with predictable monthly income, which is a stronger economic foundation than campaign-by-campaign crowdfunding alone.

VI. CONCLUSION

ContribuX demonstrates that a full-stack creator funding platform can simultaneously achieve production-grade payment security and a low-friction contributor experience using contemporary JavaScript tooling. By implementing the six-property security framework formalised in [1] on top of Razorpay's PCI-DSS infrastructure, the platform avoids storing any sensitive payment data while still providing a complete audit trail. The dual AON/KIA campaign model operationalises the demand-signalling mechanism described in [2]. The Razorpay hosted checkout addresses all eight statistically significant predictors of interface satisfaction identified in [4], in particular the two strongest—security and payment method preferences. Measured latencies confirm that the cryptographic verification overhead is negligible, and the six-property security checklist is fully satisfied. The platform is released as open-source at https://github.com/ARYAKOMPALWAR/ContribuX as both a practical tool and a reference implementation for Razorpay integration in Next.js applications.

VII. FUTURE WORK

Planned extensions are: (1) replacing filesystem image storage with cloud object storage (AWS S3 or Cloudinary) for multi-instance and Vercel serverless compatibility; (2) implementing a creator analytics dashboard with contribution trend charts using Recharts, consistent with the data-visibility design patterns in [3]; (3) adding Google

and GitHub OAuth providers to NextAuth and implementing role-based access control for platform administrators; (4) integrating blockchain-based cryptocurrency payment options (Ethereum, Polygon) to serve the international creator audience and address the legal and awareness challenges identified in [3]; and (5) conducting a formal 100-participant user study using the nine-construct satisfaction instrument of [4] to quantitatively validate interface design decisions.

REFERENCES

[1] M. A. Hassan, Z. Shukur and M. K. Hasan, "An Efficient Secure Electronic Payment System for E-Commerce," Computers, vol. 9, no. 3, p. 66, 2020.

[2] A. Miglo, "Crowdfunding in a Competitive Environment," J. Risk Financial Manag., vol. 13, no. 3, p. 39, 2020.

[3] K. Khando, M. S. Islam and S. Gao, "The Emerging Technologies of Digital Payments and Associated Challenges: A Systematic Literature Review," Future Internet, vol. 15, no. 1, p. 21, 2022.

[4] O. Tounekti, A. Ruiz-Martínez and A. F. Skarmeta-Gómez, "Users' Evaluation of a New Web Browser Payment Interface for Facilitating the Use of Multiple Payment Systems," Sustainability, vol. 13, no. 9, p. 4711, 2021.

[5] A. Kompalwar, "ContribuX Source Code," GitHub, 2025. [Online]. Available: https://github.com/ARYAKOMPALWAR/ContribuX

[6] European Central Bank, Recommendations for the Security of Internet Payments, ECB, Frankfurt, 2013.

[7] Razorpay, "Payment Gateway API Documentation," Razorpay Docs, 2025. [Online]. Available: https://razorpay.com/docs

[8] Vercel Inc., "Next.js 15 App Router Documentation," 2025. [Online]. Available: https://nextjs.org/docs

[9] MongoDB Inc., "Mongoose ODM Documentation," 2025. [Online]. Available: https://mongoosejs.com

[10] B. Campbell et al., "OAuth 2.0 Authorization Framework," RFC 6749, IETF, 2012.