IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

CodeCraft: A SaaS Multi-Language Online Code Editor with Real-Time Backend and Community Snippet Library

Abhay Chandrakar, Aniket Yadav, Aryan Kumar Rai, Tanya Chauhan
Department of Computer Science & Engineering
Raj Kumar Goel Institute of Technology, Ghaziabad, India

Abstract: *The rapid growth of cloud-based development tools has transformed the way developers write, test, and share code. This paper presents CodeCraft, a full-featured Software-as-a-Service (SaaS) online code editor providing a powerful browser-based coding environment without any local setup. The platform enables multi-language code execution in real time, reusable snippet management, and community engagement. CodeCraft is built on Next.js 15 (App Router), Convex (real-time serverless backend), Monaco Editor (VS Code engine), and Clerk (authentication), with a Lemon Squeezy freemium model. Performance evaluation shows an initial page load of ~1.2 s on Vercel's edge network, Convex query response below 100 ms, and a Lighthouse performance score of 82/100. User acceptance testing with seven participants yielded an average satisfaction rating of 4.6/5.0.*

Keywords: SaaS, Online Code Editor, Monaco Editor, Convex, Real-time Backend, Code Execution, Freemium Model, Next.js, Clerk

I. INTRODUCTION

The landscape of software development has undergone a paradigm shift with the rise of cloud computing and browser-based tools. Setting up a local development environment traditionally required installing compilers, configuring IDEs, managing dependencies, and resolving cross-platform issues — a persistent barrier for students, educators, and professionals seeking a quick coding environment [1].

Online code editors such as CodePen, JSFiddle, Replit, and GitHub Codespaces have demonstrated demand for browser-based tools. However, these platforms either narrow their scope to web front-end, lack IDE-grade editing, or omit community snippet sharing [2]. CodeCraft fills this gap: a production-grade, multi-language, freemium SaaS editor combining a professional coding environment, community snippet library, and transparent monetisation in a single TypeScript platform.

1.1 Problem Statement

Key pain-points addressed: (a) context switching between separate coding and snippet tools; (b) no unified multi-language platform with community sharing; (c) lack of professional IDE features in free tiers; (d) no persistent execution history; (e) missing community-driven snippet discovery.

1.2 Objectives

Primary objectives: (1) Monaco-based multi-language editor; (2) real-time backend via Convex; (3) OAuth authentication via Clerk; (4) freemium model with Lemon Squeezy; (5) community snippet library with starring and comments; (6) per-user execution history dashboard; (7) zero-config Vercel CI/CD deployment.

DOI: 10.48175/IJARSCT-34629

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

II. RELATED WORK

2.1 Existing Online Code Editors

Table 1 compares leading online code editors. No single platform simultaneously combines a professional Monaco-based editor, multi-language free tier, community snippet library, and transparent freemium model. CodeCraft occupies this unique market position [4].

Platform	Languages	Community Snippets	Free Tier	Editor Quality
Replit	50+	Yes (Repls)	Limited compute	Basic
CodePen	HTML/CSS/JS only	Yes	Yes	Basic
JSFiddle	Web only	No	Yes	No
StackBlitz	JS/TS ecosystem	No	Freemium	VS Code-based
CodeSandbox	JS/TS, Node	No	Limited	VS Code-based
CodeCraft	10+ languages	Yes (full)	Yes (Free+Pro)	Monaco (VS Code)

Table 1: Comparison of Existing Online Code Editors

2.2 SaaS Architecture Patterns

Modern SaaS applications apply: serverless backends that scale automatically; authentication-as-a-service to reduce security complexity; freemium monetisation for organic user acquisition; CDN-delivered SSR frontends for low latency; and webhook-driven integration for asynchronous payment events [5]. CodeCraft applies all five patterns.

2.3 Real-Time Backend Technologies

Convex introduces a reactive model where live queries automatically notify subscribed clients on data changes, ACID-compliant mutations ensure consistency, and a TypeScript-first schema provides end-to-end type safety [6]. Supabase and Firebase were also evaluated; Convex was selected for superior TypeScript integration and elimination of manual cache invalidation.

2.4 Code Execution Engines

Safe code execution requires isolation via Docker containers (Replit), Firecracker microVMs (AWS Lambda), or REST-based sandboxes. CodeCraft uses the Piston API — an open-source engine supporting 100+ languages with configurable timeout and memory limits — avoiding container management overhead [7].

2.5 Authentication-as-a-Service

Clerk was selected over Auth0, NextAuth.js, and Supabase Auth for its first-class @clerk/nextjs integration, pre-built UI components, and webhook system for lifecycle sync to Convex. Every Convex function validates identity server-side via ctx.auth.getUserIdentity() [3].

III. SYSTEM DESIGN & METHODOLOGY

3.1 System Architecture

CodeCraft follows a three-tier client-server architecture adapted for serverless, cloud-native deployment (Fig. 1). The Presentation Layer (Next.js 15, React 19, Tailwind CSS v4, Vercel edge) handles SSR. The Application Layer (Convex

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

serverless, Clerk, Piston API) encapsulates business logic. The Data Layer (Convex document DB) provides ACID transactions and real-time subscriptions.

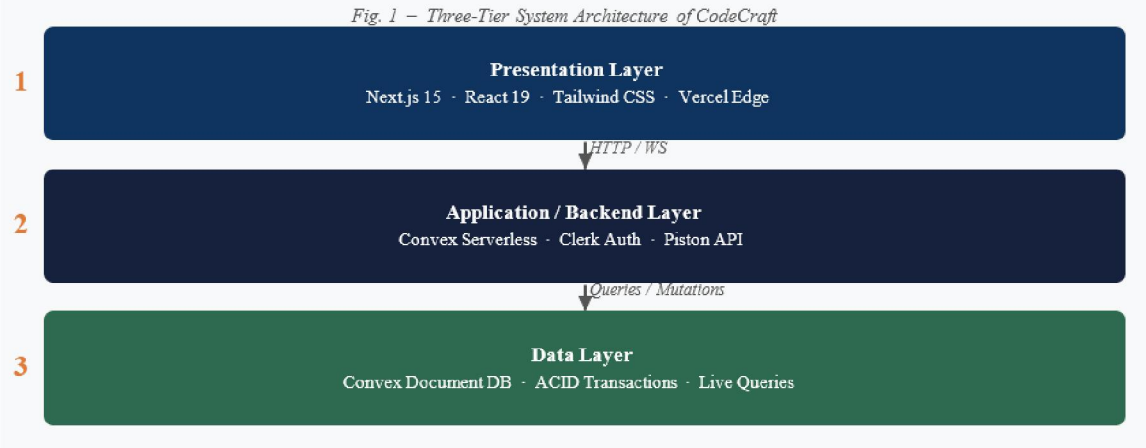
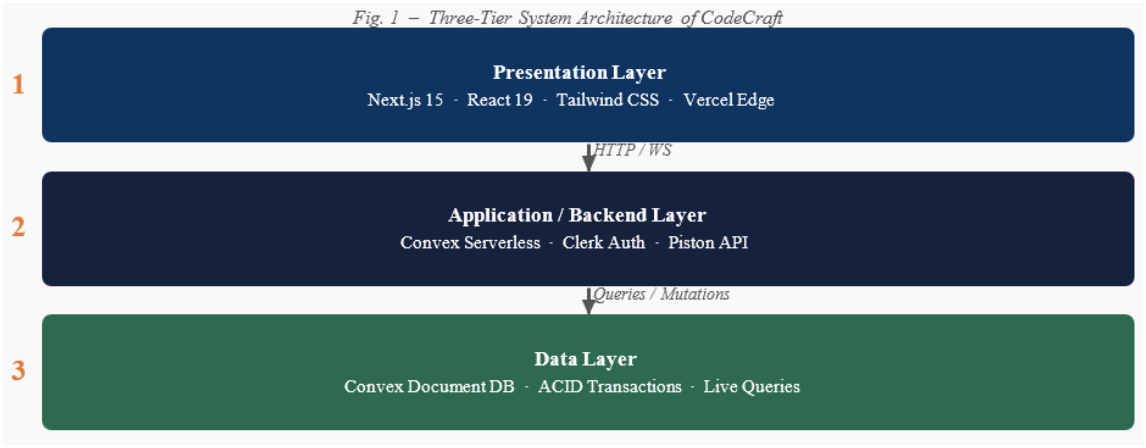


Fig. 1 – Three-Tier System Architecture of CodeCraft



The execution data flow (Fig. 2): (1) user writes code in Monaco Editor; (2) 'Run' triggers a Convex action; (3) action validates auth and Pro status; (4) POST to Piston API; (5) Piston returns stdout/stderr;
(6) result persisted to Convex; (7) response displayed in the output panel.

3.2 Technology Stack

Category	Technology	Purpose
Frontend Framework	Next.js 15 (App Router)	SSR, routing, API routes
UI Library	React 19	Component-based UI
Styling	Tailwind CSS v4	Utility-first styling
Animation	Framer Motion 12	UI transitions & effects

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

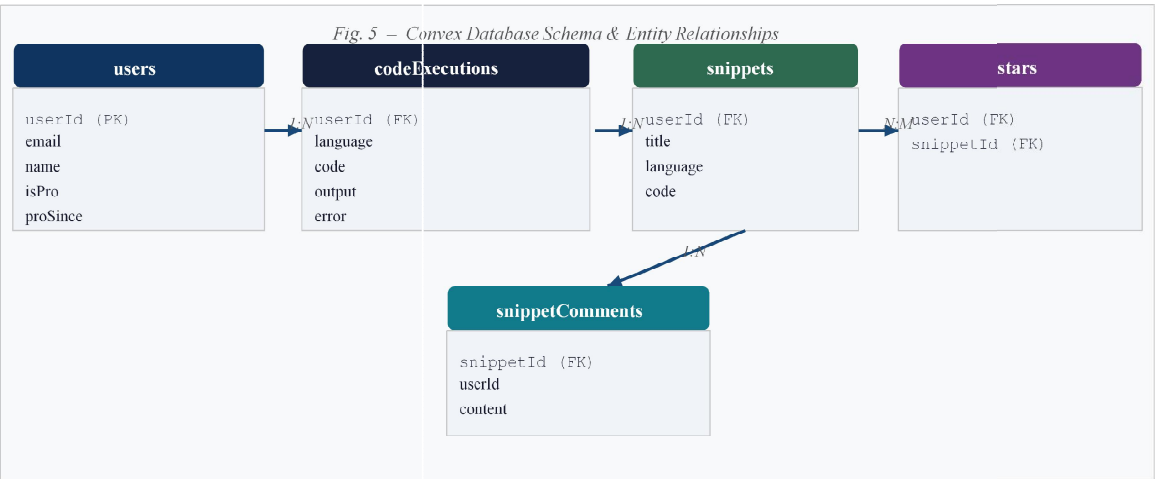
Code Editor	Monaco Editor	VS Code-like editing
Backend / Database	Convex	Serverless + real-time DB
Authentication	Clerk	OAuth, JWT, webhooks
State Management	Zustand 5	Client-side state
Code Execution	Piston API	Multi-language sandbox
Payments	Lemon Squeezy	SaaS subscription billing
Webhook Verification	Svix	Signature validation
Deployment	Vercel	Edge hosting + CI/CD
Language	TypeScript 5	End-to-end type safety

Table 2: Technology Stack Summary

3.3 Database Design

CodeCraft uses Convex's TypeScript schema definition, providing compile-time type safety for all database operations (Fig. 5). Strategic indexes — userId on users (O(1) lookup), compound (userId, snippetId) on stars (uniqueness enforcement) — ensure query performance at scale [6].

Fig. 5 – Convex Database Schema & Entity Relationships



Collection	Key Fields	Purpose
users	userId, email, name, isPro, proSince	Registered users & Pro status
codeExecutions	userId, language, code, output, error	Code run history per user
snippets	userId, title, language, code, userName	Community-published snippets

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

snippetComments	snippetId, userId, userName, content	Comments on snippets
stars	userId, snippetId	Star tracking (unique pair)

Table 3: Convex Collections (Database Schema)

3.4 Freemium Model Design

Feature	Free Tier	Pro Tier
Languages Available	JavaScript, Python	10+ languages
Code Execution	Unlimited	Unlimited
Snippet Publishing	Yes	Yes
Stars & Comments	Yes	Yes
Theme Selection	3 themes	All themes
Font Size Options	Default only	Full range (12-20 px)
Priority Support	No	Yes
Price	Free forever	Subscription / One-time

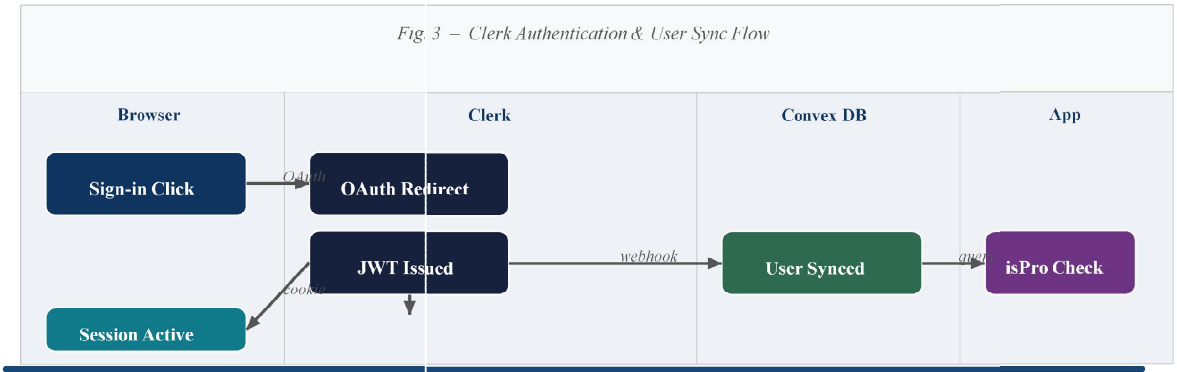
Table 4: SaaS Tier Feature Comparison (Free vs Pro)

IV. IMPLEMENTATION

4.1 Authentication Module

Authentication uses Clerk with clerkMiddleware() for Next.js route protection. A webhook endpoint syncs user.created events to Convex via Svix signature verification (Fig. 3). The isPro field is enforced server-side in every Convex function. Listing 1 shows the webhook handler.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2




```
// Listing 1 – Clerk Webhook Handler (app/api/webhooks/clerk/route.ts)
import { Webhook } from 'svix';
import { ConvexHttpClient } from 'convex/browser';
const convex = new ConvexHttpClient(process.env.NEXT_PUBLIC_CONVEX_URL!);
export async function POST
(req: Request) {
const WEBHOOK_SECRET = process.env.CLERK_WEBHOOK_SECRET!;
const body = await req.text();
const wh = new Webhook(WEBHOOK_SECRET);
let evt: WebhookEvent;
try {
evt = wh.verify(body, {
'svix-id': req.headers.get('svix-id')!,
'svix-timestamp': req.headers.get('svix-timestamp')!,
'svix-signature': req.headers.get('svix-signature')!,
}) as WebhookEvent;
} catch {
return new Response('Bad signature', { status: 400 });
}
if (evt.type === 'user.created') {
await convex.mutation(api.users.syncUser, {
userId: evt.data.id,
email: evt.data.email_addresses[0].email_address,
name: `${evt.data.first_name} ${evt.data.last_name}`,
});
}
return new Response('OK', { status: 200 });
}
```

Listing 1 – Clerk Webhook Handler

4.2 Code Editor Module

The editor uses Monaco Editor via @monaco-editor/react — the same engine as VS Code. A Zustand store (useCodeEditorStore) manages language, code, theme, font size, and output state, persisting to localStorage. The language selector is filtered by the user's isPro status fetched from Convex. Listing 2 shows the store definition.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

```
// Listing 2 – Zustand Editor Store (store/useCodeEditorStore.ts) import
{ create } from 'zustand'; import { LANGUAGE_VERSIONS, DEFAULT_CODE }
from '@/lib/constants'; interface EditorState { language: string; code:
string; theme: string; fontSize: number; output: string; isRunning:
boolean; setLanguage: (lang: string) => void; setCode: (code: string) =>
void; runCode: () => Promise; } export const useCodeEditorStore =
create((set, get) => ({ language: localStorage.getItem('lang') ??
'javascript', code: localStorage.getItem('code') ??
DEFAULT_CODE['javascript'], theme: localStorage.getItem('theme') ?? 'vs-
dark', fontSize: Number(localStorage.getItem('fontSize')) || 14, output:
'', isRunning: false, setLanguage: (lang) => {
localStorage.setItem('lang', lang); set({ language: lang, code:
DEFAULT_CODE[lang] }); }, setCode: (code) => {
localStorage.setItem('code', code); set({ code }); }, runCode: async ()
=> { set({ isRunning: true, output: '' }); // calls executeCode Convex
action – see Listing 3 set({ isRunning: false }); },
}));
```

Listing 2 – Zustand Editor Store (TypeScript)

4.3 Code Execution Module

A Convex action (executeCode) validates authentication and language entitlement, then POSTs to the Piston API. Both stdout and stderr are returned to the client and persisted to the codeExecutions collection. Listing 3 shows the complete Convex action. Table 5 lists all supported languages.

```
// Listing 3 – Code Execution Convex Action (convex/codeExecutions.ts)
export const executeCode = action({ args: { language: v.string(), code:
v.string() }, handler: async (ctx, { language, code }) => { const identity =
await ctx.auth.getUserIdentity(); if (!identity) throw new Error('Not
authenticated'); const user = await ctx.runQuery(internal.users.getByUserId,
{ userId: identity.subject }); if (PRO_LANGUAGES.includes(language) &&
!user?.isPro) throw new Error('Pro subscription required for this
language'); const res = await
fetch('https://emkc.org/api/v2/piston/execute', { method: 'POST', headers: {
'Content-Type': 'application/json' }, body: JSON.stringify({ language,
version: LANGUAGE_VERSIONS[language],
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

```
files: [{ content: code }],
}), }); const data = await
res.json();
// Persist execution record to Convex DB await
ctx.runMutation(internal.codeExecutions.save, { userId:
identity.subject, language, code, output:
data.run.stdout, error: data.run.stderr, }); return {
output: data.run.stdout, error: data.run.stderr }; },
});
```

Listing 3 – Convex Code Execution Action (TypeScript)

Language	Runtime Version	Tier	Syntax Highlight
JavaScript	Node.js 18.x	Free	Yes
Python	3.10.x	Free	Yes
TypeScript	5.x (ts-node)	Pro	Yes
Java	OpenJDK 21	Pro	Yes
C++	GCC 12	Pro	Yes
C	GCC 12	Pro	Yes
Go	1.21	Pro	Yes
Rust	1.73	Pro	Yes
C#	.NET 7	Pro	Yes
SQL	SQLite 3	Pro	Yes

Table 5: Supported Programming Languages

4.4 Snippets Module & Convex Schema

The snippet feed at /snippets is powered by a Convex live query — newly published snippets appear for all online users without a page refresh. Stars use a compound unique index on (userId, snippetId) to prevent duplicates. Listing 4 shows the full Convex schema definition that underpins all five collections.

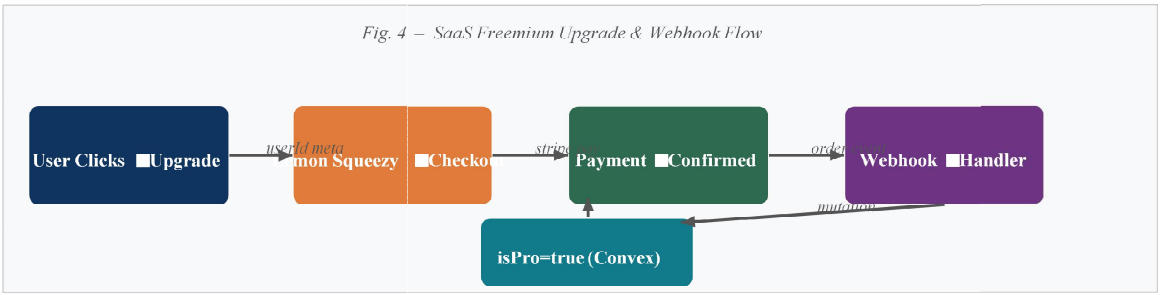
IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

```
// Listing 4 – Convex Schema Definition (convex/schema.ts)
import { defineSchema, defineTable } from 'convex/server';
import { v } from 'convex/values'; export default defineSchema({
users: defineTable({ userId: v.string(), email: v.string(),
name: v.string(), isPro: v.boolean(), proSince:
v.optional(v.number()), lemonSqueezyCustomerId:
v.optional(v.string()), }).index('by_user_id', ['userId']),
codeExecutions: defineTable({ userId: v.string(), language:
v.string(), code: v.string(), output: v.optional(v.string()),
error: v.optional(v.string()), }).index('by_user_id', ['userId']),
snippets: defineTable({ userId: v.string(), title: v.string(),
language: v.string(), code: v.string(), userName: v.string(),
}).index('by_user_id', ['userId']), stars: defineTable({ userId:
v.string(), snippetId: v.id('snippets'),
}).index('by_user_id_and_snippet_id', ['userId', 'snippetId']),
snippetComments: defineTable({ snippetId: v.id('snippets'), userId:
v.string(), userName: v.string(), content: v.string(),
}).index('by_snippet_id', ['snippetId']),
});
```

Listing 4 – Convex Schema Definition (TypeScript)

4.5 SaaS / Pricing Module

Fig. 4 illustrates the SaaS upgrade and webhook flow. Clicking 'Upgrade to Pro' generates a Lemon Squeezy checkout URL with userId in order metadata. The webhook handler verifies the HMAC signature and calls a Convex mutation setting isPro: true. Feature gating is enforced server-side in all Convex actions to prevent bypass.



4.6 User Dashboard & Deployment

The /profile page is a tabbed personal hub showing user avatar, Pro badge, aggregate statistics (total executions, snippets, stars, preferred language), paginated execution history, and published/starred snippets. Snippet deletion removes the snippet, all associated comments, and all stars in a single Convex transaction. The application is deployed to Vercel via GitHub CI/CD with Convex functions deployed via npx convex deploy, accessible at https://saas-code-editor-peach.vercel.app.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

V. RESULTS & DISCUSSION

5.1 Performance Analysis

Performance was measured on the production deployment using browser DevTools, Lighthouse, and manual timing. Table 6 summarises results. Vercel's edge CDN reduces latency significantly. The Lighthouse score of 82/100 reflects Monaco Editor's bundle size; future optimisation includes dynamic import and worker prefetching.

Metric	Measured Value	Remarks
Initial Page Load (Vercel Edge)	~1.2 s	SSR + CDN caching
Metric	**Measured Value**	**Remarks**
Monaco Editor Initialisation	~800 ms	Lazy loaded after paint
Code Execution Latency (Python)	~1.5-2.0 s	Piston API round-trip
Code Execution Latency (JS)	~1.0-1.5 s	Node.js starts faster
Convex Query Response Time	<100 ms	Real-time reactive
Snippet Feed Load (20 items)	~300 ms	Indexed Convex query
Clerk Auth Redirect	~600 ms	Standard OAuth latency
Lighthouse Performance Score	82 / 100	Monaco bundle overhead
Lighthouse Accessibility Score	91 / 100	Semantic HTML baseline
Time to Interactive (TTI)	~2.1 s	Monaco lazy load factor

Table 6: System Performance Metrics

5.2 Security Analysis

Table 7 summarises the security audit. Authentication is delegated to Clerk. All Convex functions validate identity server-side. Webhooks use Svix and HMAC signature verification. User code executes on Piston's isolated infrastructure, entirely separate from CodeCraft's application servers.

Security Dimension	Implementation	Risk Level
Authentication	Clerk OAuth + JWT	Low
Authorization	Server-side Convex validation	Low
Webhook Security	Svix + HMAC signature verification	Low
Code Execution Safety	Piston API sandboxing	Medium
Secret Management	Vercel environment variables	Low

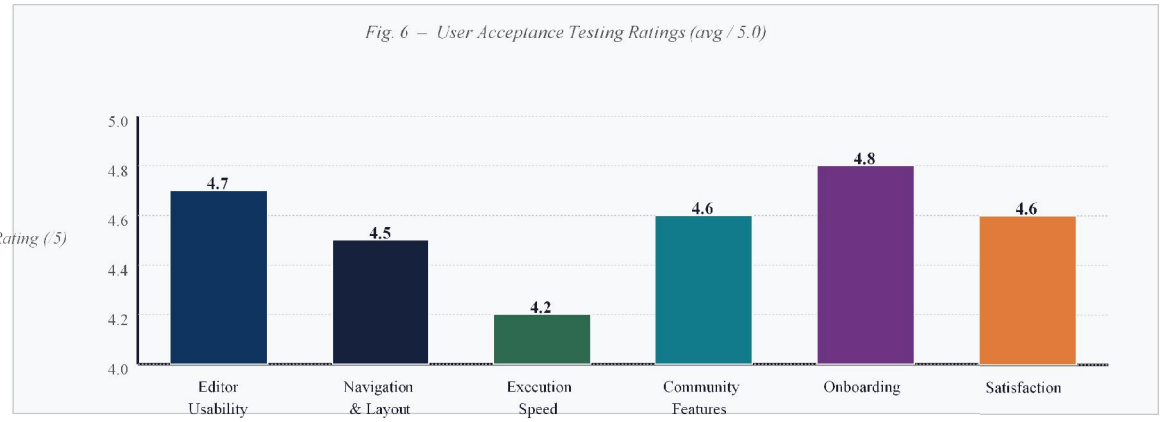
IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

NoSQL Injection	Convex schema validation	Low
XSS Prevention	React DOM escaping + CSP headers	Low

Table 7: Security Audit Findings

5.3 User Acceptance Testing

UAT was conducted with seven participants (five undergraduate CS students and two professional developers). Fig. 6 visualises the ratings. Key feedback: Ctrl+Enter execution shortcut, snippet forking, and more languages — all added to the future enhancements backlog.



Dimension	Avg. Rating (/5)	Remarks
Editor Usability	4.7	Monaco keyboard shortcuts intuitive
Navigation & Layout	4.5	Snippet filtering appreciated
Code Execution Speed	4.2	Piston latency acceptable
Community Features	4.6	Snippet feed and stars engaging
Onboarding (Sign-up)	4.8	One-click GitHub OAuth smooth
Overall Satisfaction	4.6	Strong preference over alternatives

Table 8: UAT Feedback Summary

VI. CONCLUSION & FUTURE WORK

CodeCraft successfully demonstrates a full-stack SaaS web application using a modern TypeScript stack. All objectives were achieved: Monaco-based multi-language editor, real-time Convex backend, Clerk authentication, community snippet library, and transparent freemium monetisation. The reactive nature of Convex eliminated manual WebSocket and cache management complexity. System testing confirmed all ten functional requirements; UAT yielded an average satisfaction of 4.6/5.0, validating the platform's usability and feature completeness.

Planned future enhancements: (1) Real-time collaborative editing via Y.js / Liveblocks for pair-programming; (2) AI code assistant (LLM-powered completion and debugging); (3) Ctrl+Enter execution shortcut; (4) snippet forking; (5)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

GitHub integration for direct repo push/pull; (6) containerised execution via Firecracker microVMs; (7) offline mode via service workers and IndexedDB; (8) organisation workspaces for team-shared private snippet libraries.

REFERENCES

[1] Next.js Documentation, "App Router," Vercel Inc., 2024. [Online]. Available: https://nextjs.org/docs
[2] L. C. L. Kats et al., "Software development environments on the web," in Proc. ACM Onward! 2012, pp. 99.
[3] Clerk Documentation, "Next.js Quickstart," Clerk Inc., 2024. [Online]. Available: https://clerk.com/docs[4] Monaco Editor, "The code editor that powers VS Code," Microsoft Corp., 2024.
[5] T. Ottino, "Building SaaS Products with Next.js and Convex," Medium, 2024.
[6] Convex Documentation, "Convex Backend Platform," Convex Inc., 2024. [Online]. Available: https://docs.convex.dev
[7] Piston API, "A high performance general purpose code execution engine," GitHub, 2024.
[8] Tailwind CSS, "Rapidly build modern websites," Tailwind Labs, 2024.
[9] Framer Motion, "Production-ready motion library for React," Framer, 2024.
[10] Zustand, "A small, fast and scalable state-management solution," pmndrs, 2024.
[11] Lemon Squeezy, "Payments & Subscriptions," 2024. [Online]. Available: https://www.lemonsqueezy.com
[12] Vercel, "Deploy Next.js to Vercel," 2024. [Online]. Available: https://vercel.com/docs
[13] M. Goldman, G. Little, R. C. Miller, "Real-time Collaborative Coding in a Web IDE," in Proc. ACM UIST2011, pp. 155-164.
[14] IEEE Std 829-2008, "Standard for Software and System Test Documentation," IEEE, 2008.
[15] R. Pressman, "Software Engineering: A Practitioner's Approach," 8th ed., McGraw-Hill, 2015.
[16] Svix, "Webhooks as a Service," Svix Inc., 2024. [Online]. Available: https://www.svix.com