CASE STUDY

Nemrah Ahmad LMS

Full-Stack Content Platform · React Native + Next.js · Feb–Sep 2024

Muhammad Hassan · Lead Developer · hassanprodeveloper.com

3	8 Months	100%	iOS + Android
PLATFORMS SHIPPED	TIMELINE	SCREENSHOT BLOCKED	BOTH STORES LIVE

React Native · Next.js Admin · Next.js Website · Lexical Editor · Audio Books · Deep Linking · Lead Developer

PROJECT OVERVIEW

Nemrah Ahmad is one of Pakistan's most widely read novelists. Her content was being pirated — third parties copying novels and courses and selling them without her permission. **I led the end-to-end development of a complete content platform**: a React Native mobile app (iOS + Android), a Next.js multi-role admin panel, and a Next.js marketing website — all built from scratch and shipped in 8 months.

CONTENTS

A Best-Selling Author Losing Revenue to Piracy

Nemrah Ahmad is one of Pakistan's most widely read novelists — her work commands a large, loyal audience. But that popularity came with a painful side effect: pirated copies of her novels and courses were circulating freely online. Third parties were reproducing her content, packaging it cheaply, and profiting without her knowledge. She had no technical means to stop it.

WHY THIS MATTERED

Screen-capture prevention was not a nice-to-have. It was the entire justification for building a proprietary platform. Every architecture decision flowed from this requirement.

PLATFORM REQUIREMENTS AT A GLANCE

Area	Required
Mobile App	iOS + Android, novels, courses, audiobooks, community, offline reading, DRM
Authentication	Email, Google Sign-In, Apple Sign-In with OTP verification
Reader	Paginated text, zoom, bookmarks, bilingual (Urdu/English), audio-text sync
Audiobook	Playback with auto-scrolling synced transcript highlights
Community	Gifted-chat feed, discussion boards, status updates
Payments	Stripe in-app purchases + manual payment approval flow
Admin Panel	Multi-role ACL, content upload, Lexical editor, subscription management
Website	Content listing, public playback, SEO, deep links to app
DRM	Screen capture / recording blocked on ALL paid content screens

Three Platforms, One Team, Eight Months

Before a single line of code was written, I was in the room for pre-development discussions with the backend developer, the manager, and the designer. My role was not just to execute — it was to shape what we were building. The scope: a React Native mobile app, a Next.js admin dashboard, and a Next.js marketing website, all integrated, all launched within eight months.

I had a junior developer alongside me, but the architecture decisions, technology choices, and the majority of implementation fell to me. Working with a backend developer building the API, a designer providing Figma files, and a project manager — my job was to translate all of that into functioning product and ensure the engineering side didn't become the bottleneck.

PLATFORM SCOPE

React Native App	Next.js Admin Panel	Next.js Website
iOS + Android: novels, courses, audiobooks, community, offline reading, DRM, both stores published	Multi-role ACL, lexical editor, audio-text sync, content upload, subscriptions, community moderation	Content listing, public playback, user auth, SEO-optimized pages, deep links to the mobile app

CONSTRAINTS

TIMELINE & SCOPE

- Feb–Sep 2024 (8 months hard deadline)
- Three separate repositories and codebases
- No existing codebase — greenfield from scratch
- iOS + Android (both stores, first submission)
- Urdu and English language support in reader

TEAM & RESOURCES

- 1 lead developer (me) + 1 junior developer
- 1 backend developer building the API
- 1 designer providing Figma files
- 1 project manager coordinating delivery
- Content piracy prevention was non-negotiable

MY ROLE

Lead Developer — I made every technology decision, set up project architecture across all three platforms, managed the junior developer, collaborated with backend and design, and personally built the majority of the product. I also handled both App Store and Google Play Console setup and the final store submissions.

The Six Decisions That Defined the Platform

The complexity of this project was not in any single feature — it was in making the right architectural choices under time pressure, before knowing exactly what the product would become. Here are the six decisions that shaped everything:

1

React Native CLI over Expo — deliberate loss of convenience for control

Expo would have been faster to start and easier to hand off. I chose React Native CLI instead. The core reason: react-native-screenguard requires direct access to the Android/iOS native layer. Expo's managed workflow makes this either impossible or unreliable. DRM was non-negotiable, so the build tooling had to support it without compromise.

2

RNEUI for UI — speed without sacrificing theming

With a large UI surface (reader, audiobook player, community, dashboard, tasbeeh, library) and one lead developer, I needed a component framework that handled theming and common patterns without boxing me in. RNEUI gave us a consistent design system and dark/light theming out of the box, letting the junior developer build UI screens faster while I handled complex logic layers.

3

Lexical for the admin editor — custom audio-timestamp sync nodes

The audiobook transcript sync required a rich text editor where each text segment could be annotated with a timestamp — 'highlight this sentence when audio is between 00:12 and 00:18.' No off-the-shelf editor supports this. I chose Lexical (Meta's extensible editor framework) and built a custom node type that stores time intervals per text block, powering the synchronized transcript view in the mobile reader.

4

Redux Toolkit + redux-persist — justified by real complexity

Some projects are over-engineered with Redux. Here it was justified: user auth, library state, download queue, reader preferences, community state, and tasbeeh counter all needed to survive app restarts and be accessible across deeply nested screens. redux-persist with MMKV as the storage engine gave us fast, reliable persistence without the complexity of a custom solution.

5

Screen capture prevention as a first-class architectural requirement

Most apps treat security as an afterthought. Here, content protection was the entire reason the platform was being built. react-native-screenguard was integrated from day one on both Android and iOS. Every screen displaying paid content has this active. It was tested on real devices on both platforms before any other feature was considered complete.

6

Deep linking with custom domain — bridging web to app seamlessly

When a user taps a novel on the marketing website and the app is installed, they should land directly on that novel's screen. I implemented universal links (iOS) and app links (Android) using a custom domain, with React Navigation handling in-app routing. This closed the conversion loop from website discovery to in-app engagement.

What Was Built

I set up the full project architecture across all three platforms from blank repositories, made all technology decisions, managed the junior developer's workload, and personally implemented every complex feature. Here is what shipped.

REACT NATIVE MOBILE APP

React Native CLI · TypeScript · Feb–Sep 2024

✓ Project setup, architecture, navigation (Bottom Tabs + Stack)	✓ Auth: email, Google Sign-In, Apple Sign-In + OTP
✓ Dashboard: daily goals, streak tracking, navigation shortcuts	✓ Library: personal shelf, reading history, offline downloads
✓ Reader: paginated text, zoom, bookmarks, Urdu/English support	✓ Audiobook player with synced transcript (react-native-track-player)
✓ Community feed (gifted-chat), discussion boards, status updates	✓ In-app purchases: Stripe + react-native-iap + manual approval flow
✓ Push notifications via Firebase + custom event support	✓ Deep linking: custom domain ⟶ in-app screen routing
✓ Screen-capture prevention on all paid content screens	✓ Offline chapter reading via react-native-fs file system
✓ Tasbeeh counter with admin content + push on update	✓ Avatar picker, download manager, issue reporting, boot splash

NEXT.JS ADMIN PANEL

Next.js 13 · MUI · CASL · Lexical · AWS S3

✓ Multi-role ACL using CASL (admin, moderator, content manager)	✓ Lexical editor + custom audio-timestamp nodes for transcript sync
✓ Content upload: novels, courses, chapters with rich metadata	✓ Audio management: per-paragraph time interval configuration
✓ Community moderation + reply as Nemrah Ahmad	✓ Subscription management: active users, manual payment approval
✓ Tasbeeh management + push notifications on update	✓ Daily goal configuration (streak threshold)
✓ User management: profiles, suspend, device details	✓ Custom push event sender: target user segments
✓ AWS S3 integration for all media assets	✓ Analytics: active users, subscriptions, content engagement

NEXT.JS MARKETING WEBSITE

Next.js 14 · Tailwind CSS · MUI · next-auth · Redux

✓ Content listing: all novels and courses with metadata	✓ Public content playback (non-DRM chapters, trailers)
✓ User authentication via next-auth	✓ Deep link generation: web novel pages ⟶ app screens
✓ Responsive Tailwind CSS + MUI design	✓ SEO-optimized pages for novel and course discovery

8-Month Sprint Breakdown

Period	Focus	Deliverables
Feb 2024	**Discovery & Architecture**	Pre-dev meetings, technology decisions locked, all three repos set up, architecture and navigation structure defined.
Mar–Apr 2024	**Core Mobile App**	Authentication, dashboard, library, reader. Screen-capture prevention verified on real devices. Offline chapter download with file system management.
May–Jun 2024	**Audiobooks + Community + Admin V1**	Audio player with synced transcript (custom Lexical nodes). Community feed, discussion boards, status. Admin V1: content upload, user management, ACL.
Jul 2024	**Payments + Notifications + Deep Links**	Stripe + in-app purchase integration. Manual payment approval flow. Firebase push with custom events. Deep linking via custom domain.
Aug 2024	**Website + Admin Polish + QA**	Next.js website built and integrated. Admin ACL refinement, subscription management. Full QA across Android and iOS real devices.
Sep 2024	**App Store & Play Store Launch**	Google Play Console + App Store Connect setup. Release builds, store listings written, both review processes passed first attempt. App went live.

Key Dependencies & Rationale

Package	Version	Rationale
react-native (CLI)	**0.73.9**	Full native module access required for DRM (screen-capture prevention)
react-native-screenguard	**1.0.2**	Core DRM: screen capture + recording prevention on Android and iOS
@rneui/themed	**edge**	UI framework with dark/light theming for faster multi-screen development
redux-toolkit + redux-persist + mmkv	**latest**	Complex cross-screen state with fast synchronous MMKV persistence
react-native-track-player	**4.1.1**	Background audio playback for audiobooks with custom transcript sync
@react-native-firebase/messaging	**20.3**	Push notifications and custom event payloads
@stripe/stripe-react-native	**0.37.3**	In-app purchases and subscription billing
lexical + @lexical/react	**0.14.3**	Extended with custom audio-timestamp nodes for admin transcript editing
@casl/ability + @casl/react	**6.3.3**	Multi-role access control across admin panel
@aws-sdk/client-s3	**3.550**	Media storage for novel chapters and audio assets
react-native-fs (dr.pogodin)	**2.27.1**	Offline chapter downloads to device file system
react-native-iap	**12.15.1**	Cross-platform in-app purchase alongside Stripe
next (Admin)	**13.2**	Admin dashboard — App Router, MUI, CASL, Lexical
next (Website)	**14.2**	Marketing website — App Router, Tailwind, next-auth, Redux

Shipped, Live, and Protecting Her Content

3	100%	2	8 Months
PLATFORMS DELIVERED	SCREENSHOT BLOCKED	STORES LAUNCHED	FULL DELIVERY
App · Admin · Website	*All paid screens, iOS + Android*	*Both passed review first attempt*	*Feb–Sep 2024, complex scope met*

QUALITATIVE WINS

Win	Detail
Content control reclaimed	Nemrah Ahmad now controls distribution end-to-end — no third-party platform dependency or revenue leakage.
Piracy vector closed	Screen-capture prevention blocks copying via screenshot or recording on both Android and iOS.
Community engagement	Fans message Nemrah Ahmad directly; her replies are visible to all — a unique mechanic that drives retention.
Offline access	Per-chapter downloads give users access without connectivity — important for the mobile-first Pakistani audience.
Audio-synced transcript	Text highlights word-by-word as the audio plays, matching premium audiobook app UX.
Flexible payments	Manual payment approval lets the admin team onboard bank-transfer users without international cards.
Web-to-app conversion	Deep linking closes the discovery-to-engagement loop: a novel on the website opens directly in the app.

What This Project Proved

Decision-making under uncertainty

This was the most scope I had ever owned end-to-end. Three platforms, eight months, one lead developer. The moments that tested me most were not the complex features — they were the decisions made under ambiguity before development started. Choosing React Native CLI over Expo, extending Lexical rather than building a custom editor, structuring the Redux store to survive across all three apps — these choices made or broke the timeline.

Managing a junior developer

Delegating UI screens while retaining oversight on logic integration meant I had to stay aware of two levels of complexity simultaneously. In hindsight, I would invest more time in a shared technical onboarding document so the junior could be unblocked without always needing me. The absence of clear delegation guidelines created unnecessary bottlenecks in weeks 3–4.

DRM as a first-class requirement

The screen-capture prevention feature is the one I'm most proud of — not because it was technically the hardest, but because it was the most important to the client and the one most developers would have treated as an edge case. I treated it as a first-class requirement from day one, integrated before the feature set was finalized.

What I'd do differently

- Earlier automated tests for the reader and audio-sync logic — bugs there were expensive to trace late in the project.
- A shared component library between the admin panel and website from the start, to avoid UI duplication.
- More granular feature flags in the admin panel so new content types could roll out without a full redeploy.

> *“The most valuable thing I built wasn’t a feature — it was the system that made a novelist’s work impossible to steal.”*
>
> — Muhammad Hassan · hassanprodeveloper.com

Got a platform that needs to be built right?

I specialize in React Native, full-stack Next.js, and building complete product ecosystems from discovery to App Store launch.

hassanprodeveloper.com
hassanprodeveloper@gmail.com
linkedin.com/in/hassanprodeveloper
github.com/hassanprodeveloper