# Tachi Protocol: A Strategic and Technical Execution Plan

## Part I: Strategic Due Diligence and Risk Mitigation

This initial part of the report provides a rigorous validation of the core business opportunity presented by the Tachi protocol. It begins by affirming the venture's foundational strengths as identified in the provided research, establishing a clear understanding of the market dynamics and strategic positioning. Subsequently, it conducts a critical analysis of the primary challenges and execution risks, proposing concrete, multi-layered mitigation strategies. This strategic due diligence serves as the essential groundwork upon which the detailed technical execution plan is built.

### Section 1: Affirming the Tachi Opportunity: A Synthesis of Market Validation

The Tachi protocol is positioned to address a profound and escalating market failure within the digital content ecosystem. A synthesis of the available market analysis confirms the venture's foundational strengths, which are rooted in a quantifiable problem, advantageous market timing, and a defensible architectural moat.

#### 1.1 The Problem is Real, Urgent, and Quantifiable

The long-standing, symbiotic relationship between content publishers and information indexers has been fundamentally broken by the rise of large language models (LLMs).[1] The core of the problem is a shift from a model of traffic referral to one of uncompensated data harvesting, creating an existential threat for content creators.[1]
The financial impact of this shift is staggering. Media publishers are estimated to lose over $2.3 billion annually in advertising revenue due to AI bots scraping content without providing reciprocal value in the form of user traffic.[1] This is not a theoretical loss but a direct consequence of a dramatic asymmetry in value exchange. Data from mid-2025 illustrates this disparity with stark clarity: while Google's crawler referred one human visitor for every 14 pages it scraped, OpenAI's ratio was a far more extractive 1-to-1,700. Anthropic's crawler was

even more aggressive, with a ratio of 1-to-73,000.[1] One major sports website reported receiving 13 million bot visits in a single month, which resulted in only 600 human referrals—a relationship described as "virtually parasitic".[1] This dynamic directly erodes publisher revenue by eliminating ad impressions and subscription conversion opportunities, while simultaneously devaluing the original content by using it to power AI products that disintermediate the publisher from their audience.[1]

The most compelling evidence of this acute pain is behavioral. A remarkable 73% of publishers have resorted to the drastic measure of actively blocking known AI crawlers via their robots.txt files or more advanced firewall rules.[1] This action, which risks sacrificing potential search visibility and future AI-driven reach, demonstrates a clear preference for forgoing potential benefits over tolerating uncompensated value extraction. This widespread adoption of blocking tactics is a powerful leading indicator of latent demand for a viable monetization framework. Publishers are not passively accepting the new status quo; they are actively seeking solutions that allow them to control and monetize access to their core assets.

This market turmoil has given rise to what is being termed an "AI consent crisis".[1] With over 70 lawsuits filed against major AI firms like OpenAI, Meta, and Microsoft by a growing coalition of news publishers, authors, and artists, the legal landscape is fraught with uncertainty.[1] These legal battles, initiated by major outlets such as The New York Times, highlight the industry's view that current AI scraping practices constitute copyright infringement and violate terms of service.[1] This unstable legal footing creates significant risk for both publishers and AI developers, underscoring the urgent need for a clear, scalable, and legally sound framework for content licensing—the very framework Tachi aims to provide.

### 1.2 Market Timing and Regulatory Tailwinds

The Tachi protocol is entering the market at a uniquely opportune moment, where external pressures are aligning to create a strong demand for a permissioned data access solution. The most significant of these is the shifting regulatory landscape, particularly in Europe.

The European Union's AI Act, with enforcement expected around 2026, is poised to be a "game-changer" for the industry.[1] The Act imposes stringent requirements for data governance, quality, and legal compliance, especially for systems classified as "high-risk AI".[1] Article 10 of the Act, for example, mandates that such systems be trained on datasets subject to robust governance practices, including thorough documentation of their provenance.[1] This creates a significant compliance burden for AI companies that have historically relied on indiscriminately scraped web data of unknown origin. Tachi's architecture, with its on-chain, immutable "Proof-of-Crawl" ledger, is perfectly positioned to serve as a "compliance-in-a-box" solution.[1] It provides AI developers with a transparent, auditable trail of their data sourcing, directly addressing the Act's requirements and offering a powerful tool to de-risk their operations in the EU market.

Furthermore, the emergence of direct competitors serves as a powerful form of market validation. The launch of "Pay-Per-Crawl" marketplaces by infrastructure giant Cloudflare and the venture-backed startup TollBit confirms that the concept of a transactional access model is gaining traction on both sides of the market.[1] These platforms, which have already signed up thousands of publishers, demonstrate that the industry is actively moving beyond the simplistic "allow all or block all" binary and is exploring more nuanced, economic solutions.[1] While these incumbents represent a competitive threat, their presence validates Tachi's core premise and helps educate the market, creating a fertile ground for a more advanced, decentralized alternative to take root.

### 1.3 The Strategic Moat of Decentralization

Tachi's most durable competitive advantage lies not in a specific feature but in its fundamental architecture and philosophy. While incumbents are building centralized, proprietary toll roads, Tachi is designed to be an open, credibly neutral, and decentralized public highway system.

The provided SWOT analysis correctly identifies this open standard as Tachi's key differentiator against the closed ecosystems of its competitors.[1] Cloudflare's model, while powerful due to its massive distribution, is inherently a walled garden; both publishers and AI crawlers must operate within the Cloudflare ecosystem, using its proprietary authentication and fiat-based payment processing.[1] TollBit, as a centralized startup, acts as a traditional intermediary, brokering deals and taking a cut.[1]

Tachi's architecture, based on open protocols like DIDs for identity and on-chain smart contracts for payment and audit, offers a fundamentally different value proposition. It enables direct, peer-to-contract interactions without a central gatekeeper. This model is not easily replicated by incumbents without them fundamentally undermining their own business models. For Cloudflare to adopt a truly open, decentralized payment rail, it would mean ceding its lucrative position as the central "Merchant of Record".[1] For TollBit, it would mean obviating its role as a necessary intermediary. This dynamic, often referred to as the "innovator's dilemma," creates a strategic moat for Tachi.

The defensibility of the protocol, therefore, comes from its potential to become a universal standard—a piece of public infrastructure that any publisher, CDN, or AI company can integrate with, rather than a proprietary service they must subscribe to. This openness fosters a more resilient and composable ecosystem, preventing vendor lock-in and aligning with the core ethos of the open web.[1]

## Section 2: Critical Analysis of Core Challenges & Mitigation Strategies

While the Tachi protocol is built on a solid strategic foundation, its success is contingent upon navigating several significant execution risks. The provided research materials correctly identify the "cold start" problem, the legal novelty of the licensing model, and the competitive threat from incumbents as primary challenges.[1] This section moves beyond identification to offer a critical analysis of these risks and propose concrete, multi-layered mitigation plans to address them.

### 2.1 The Cold Start Problem: Architecting the "Atomic Network"

The most formidable challenge for any two-sided marketplace is overcoming the "cold start" or "chicken-or-egg" problem: AI companies will not adopt the protocol without a critical mass of valuable content, and publishers will not see the value in adopting it without paying AI crawlers.[1] The proposed strategy of constraining the market to a high-value "atomic network" is the correct approach, but its success depends entirely on the selection of that initial network.[1]

The leverage against AI crawlers is not uniform across all types of content. Generic news reports, "how-to" guides, and basic informational articles are largely commodities. If one source for this type of content is placed behind a paywall, an AI model can likely find a similar, free alternative elsewhere with minimal loss in quality. The key to creating leverage is to aggregate content that is high-value, time-sensitive, and non-commoditized. This is content for which AI models have no viable, free substitute.

The research's suggestion to target "financial analysis newsletters" is precisely the right path.[1] High-quality financial data and analysis are extremely valuable for training and operating specialized financial AI models. The value of this information decays rapidly, making real-time, up-to-the-minute access essential. Furthermore, the unique analysis and proprietary insights from top-tier sources cannot be easily replicated. This combination of high value, time sensitivity, and non-commoditization gives publishers in this vertical significant pricing power and leverage.

A concrete mitigation strategy is therefore to execute a go-to-market plan that is laser-focused on creating an "atomic network" within this specific vertical. Based on an analysis of leading financial and business newsletters, an initial target list can be formulated.[3] The primary goal should be to form a coalition of 5 to 10 top-tier publications from this list (e.g., Axios Pro Rata, The Daily Upside, select premium newsletters from Bloomberg and Reuters) for the initial pilot program.

Secondary and tertiary targets should follow the same logic of high-value, non-commoditized content. This includes niche, high-authority technology blogs that provide deep technical analysis valuable for training coding assistants and other specialized models.[8] Another promising vertical is open-access scientific journals, whose peer-reviewed research is an invaluable and irreplaceable data source for scientific and medical AI applications.[10] By successfully cornering a "must-have" dataset in one of these niches, Tachi can create a gravitational pull that forces the relevant AI companies to the table, thus kickstarting the adoption flywheel.

### 2.2 Legal Novelty: From Theoretical Soundness to Operational Readiness

The research correctly argues that the protocol's licensing mechanism—combining a Crawl-NFT linked to terms of service with the act of an on-chain payment—is likely to be legally enforceable under existing electronic contract law frameworks like the E-SIGN Act in the US.[1] However, a critical shortcoming in this analysis is the underestimation of the *operational friction* that this legal novelty will create during the sales and onboarding process with enterprise-level publishers.

The primary risk is not a future adverse court ruling, but the immediate, practical barrier of convincing a risk-averse corporate legal department to approve a novel, crypto-based contract formation structure. A publisher's general counsel will not be satisfied with the assertion that the model is "probably enforceable." They will require extensive, time-consuming, and costly reviews, which can easily stall or kill a potential partnership. This legal friction is a sales and product marketing challenge as much as it is a legal one.

To mitigate this, Tachi must proactively dismantle these barriers. The strategy should involve three key pillars:

1. **Commission Formal Legal Opinions:** The first step is to engage top-tier legal counsel with expertise in technology, contract, and intellectual property law in key jurisdictions (primarily the US and EU). The objective is to produce formal, written opinion letters that analyze the enforceability of the protocol's on-chain licensing mechanism. These documents are not for internal reassurance; they are powerful sales enablement tools that can be provided directly to the legal teams of prospective publisher partners, dramatically accelerating their review process and building confidence.
2. **Productize the Legal Framework:** The front-end MVP must be designed to make the legal flow transparent and intuitive for the publisher. The onboarding process should not just be a series of technical steps; it should clearly articulate the formation of a legal agreement. This means including a dedicated step in the onboarding wizard, perhaps titled "Review Terms & Create License," which presents a plain-English summary of the Offer-Acceptance-Consideration flow. This screen should clearly state that by minting the NFT, the publisher is making a formal offer to license their content under the specified terms, which are immutably linked via an IPFS hash.
3. **Draft a Bridging Master Service Agreement (MSA):** To accommodate the standard operating procedures of large media organizations, Tachi should develop a traditional, off-chain MSA. This document would be signed by early publisher partners and would serve to bridge the gap between their existing legal processes and the novel on-chain protocol. The MSA would explicitly reference and incorporate the on-chain licensing mechanism, providing a familiar legal wrapper around the new technology and giving corporate legal teams a document they recognize and can work with.

### 2.3 The Competitive Threat: Neutralizing the Incumbent's Distribution Advantage

The research correctly highlights Cloudflare's massive distribution network—serving roughly 20% of the web—as a formidable competitive advantage.[1] The primary threat is not that

Cloudflare will build a technologically superior product, but that it will leverage this distribution to offer a "good enough," centralized, fiat-based solution that is bundled for free or at a very low cost with its core CDN and security services.

The deeper threat lies in Cloudflare's demonstrated power to shape the web's default behaviors. By enabling default AI bot blocking for new domains, Cloudflare has shown it can influence the actions of millions of website owners with a single policy change.[1] Their strategic advantage is the ability to make their solution the "easy button." This could position Tachi's more robust, sovereign, and economically aligned solution as being overly complex and unnecessary for the average user, thereby capturing the market before Tachi can achieve critical network effects.

Tachi cannot win a head-to-head battle on distribution. Therefore, its strategy must be to compete on philosophy, value alignment, and superior economics. The go-to-market narrative must be sharply focused on the principles of **data sovereignty, the absence of vendor lock-in, and fair, transparent economics.** The marketing and business development efforts should specifically target publishers who are already wary of ceding more control and revenue to the large, centralized gatekeepers of the web, such as Google and Cloudflare.[1] Many publishers already feel that their current predicament is a direct result of their dependence on these platforms.[1]

This strategy turns Tachi's architectural difference into its most potent marketing weapon. It is not just another tool; it is the technological backbone for a movement. The goal is to build a coalition of the ideologically aligned, using the promise of a truly open, publisher-first standard as a powerful rallying cry. This approach reframes the choice for publishers: it is not merely a choice between two competing products, but a choice between perpetuating dependence on a centralized gatekeeper or embracing an open standard that restores their sovereignty.

The following table summarizes these primary strategic risks and the proposed mitigation plans, providing a clear, at-a-glance action plan.

| Risk Category | Risk Description | Primary Mitigation Strategy | Secondary/Tactical Mitigations | Key Success Metric |
|---|---|---|---|---|
| **Marketplace Cold Start** | Inability to attract initial publishers and AI crawlers to create a viable, liquid market. [1] | Execute a constrained go-to-market strategy by targeting a high-value "atomic network" of non-commoditized content publishers. [1] | 1. Focus initial outreach on 5-10 top-tier financial newsletters and high-authority tech blogs.<br>2. Provide "white-glove" onboarding support and gas fee subsidies for the first 20 | Onboard at least 5 publishers from a pre-defined list of top 50 high-value targets within the first 6 months. |

|  |  |  |  |  |
|---|---|---|---|---|
|  |  |  | publishers.<br>3. Develop public case studies from the atomic network to attract the next wave of adopters. |  |
| **Legal Novelty** | Operational friction and sales cycle delays caused by the untested legal nature of the on-chain licensing model. [1] | Proactively de-risk the legal component by commissioning formal legal opinions and "productizing" the legal framework within the MVP. | 1. Engage top-tier legal counsel in the US/EU to produce formal opinion letters on enforceability.<br>2. Design the MVP onboarding flow to clearly explain the contract formation process.<br>3. Create a standard off-chain MSA to bridge the gap for enterprise partners. | Secure a signed letter of intent (LOI) or pilot agreement from a major, publicly-traded media company within 9 months. |
| **Competitive Reaction** | Incumbents like Cloudflare leverage their massive distribution to offer a "good enough" centralized alternative, stifling Tachi's growth. [1] | Compete on philosophy and value alignment, positioning Tachi as the open, sovereign alternative to centralized, proprietary gatekeepers. | 1. Focus marketing messaging on "data sovereignty," "no vendor lock-in," and "superior economics."<br>2. Target publishers who have publicly expressed concerns about the power of Big Tech platforms.<br>3. Open-source key components of the protocol to foster community trust and | Achieve public endorsement or adoption from a recognized publisher industry group (e.g., News/Media Alliance, IAB Tech Lab). |

| | | | | |
|---|---|---|---|---|
| | | | contributions. | |

# Part II: The Tachi Protocol: A Step-by-Step MVP Execution Plan

This part of the report translates the high-level strategic and architectural vision into a granular, actionable development plan. The primary focus is on the creation of a Minimum Viable Product (MVP), specifically the publisher-facing front-end application designed to serve as a powerful demonstration tool for securing investment and publisher partnerships. The plan is structured to be executed by a technically proficient founder, leveraging modern development tools and AI coding assistance.

## Section 3: Foundational MVP Technology Stack

The selection of the technology stack is a critical decision that will dictate development velocity, scalability, and the overall developer experience. The choices outlined here are optimized for rapid prototyping, security, and alignment with the modern Web3 ecosystem, directly supporting the MVP's goals.

### 3.1 Publisher Front-End

- **Framework: Next.js (App Router):** Next.js is the definitive choice for the publisher dashboard. Its App Router architecture provides a powerful combination of React Server Components (RSCs) for fast initial page loads and SEO-friendly static content, and Client Components for rich, client-side interactivity.[15] This hybrid model is perfectly suited for a dashboard application, where some elements (like layout and static text) can be rendered on the server, while others (like interactive forms and data displays) require client-side logic. The framework's intuitive file-based routing system significantly accelerates development, and its seamless, zero-configuration deployment to the Vercel platform is a critical advantage for a startup prioritizing speed and operational simplicity.[15]
- **Styling: Tailwind CSS:** To facilitate rapid UI development, Tailwind CSS will be utilized. As a utility-first framework, it allows for the construction of complex and custom designs directly within the markup, eliminating the need to write and maintain separate CSS files.[17] This approach dramatically speeds up the process of building and iterating on components. The framework is highly customizable through its tailwind.config.js file and the @theme directive, enabling Tachi to establish a unique and consistent brand identity across the application while leveraging a robust and

well-documented design system.[18]

- **Component Library: Shadcn/UI:** Rather than adopting a traditional, monolithic component library, the MVP will use Shadcn/UI. This is not a library in the typical sense, but a curated collection of beautifully designed, accessible, and reusable components that are copied directly into the project's codebase via a CLI command.[20] This approach aligns perfectly with Tachi's philosophy of control and openness. It provides the team with full ownership over the component code, making customization and extension trivial, while avoiding the dependency bloat of traditional libraries. The components are built using Tailwind CSS for styling and Radix UI for accessibility and unstyled primitives, ensuring a high-quality foundation.[20]

### 3.2 Web3 Stack

- **Wallet Connection: RainbowKit:** For handling wallet connections, RainbowKit is the industry-leading solution. It provides a best-in-class, highly customizable, and user-friendly "Connect Wallet" modal that handles the complexities of wallet management, network switching, balance display, and ENS address resolution out of the box.[22] Built on top of the Wagmi library, it offers a seamless and polished user experience that is critical for onboarding publishers who may be less familiar with Web3.[24]
- **Blockchain Interaction: Wagmi:** Wagmi will serve as the primary interface between the Next.js front-end and the blockchain. It is a comprehensive library of React Hooks that abstracts away the complexity of interacting with Ethereum smart contracts.[25] Using hooks like
  useAccount, useConnect, useReadContract, and useWriteContract, the development of on-chain interactions becomes declarative and integrates cleanly into the React component model. The MVP will utilize Wagmi v2, which relies on Viem as its core dependency, ensuring a modern and performant stack.[26]
- **Core Library: Viem:** Viem is a modern, lightweight, and highly performant TypeScript interface for Ethereum that has become the new standard for client-side blockchain interaction.[27] As the underlying engine for both Wagmi and RainbowKit, its inclusion ensures a cohesive and efficient stack. Viem's key advantages include its superior tree-shakability, which helps keep the final application bundle size small, and its rigorous TypeScript support, which provides enhanced type safety and auto-completion, leading to a more robust and maintainable codebase.[1]

### 3.3 Backend and Protocol Stack

The backend and on-chain components of the protocol will be built on the robust foundation outlined in the detailed technical plan.[1]

- **Blockchain: Base L2:** The choice of Base, an Ethereum Layer 2 network incubated by Coinbase, is confirmed as optimal. Its primary advantages are its ultra-low transaction fees (projected at less than $0.0001 per transaction), which are essential for the economic viability of micropayments, its full EVM compatibility, which provides access to a mature tooling ecosystem, and its deep integration with the Coinbase ecosystem, which offers robust liquidity for the USDC stablecoin and trusted on/off-ramps for publishers.[1]
- **Smart Contracts: Solidity with a Hybrid Foundry & Hardhat Workflow:** The smart contracts will be written in Solidity, the most mature language for the EVM.[1] The development process will employ a sophisticated hybrid workflow that leverages the distinct strengths of the two leading development frameworks. Foundry will be used for core contract development and unit testing due to its exceptional speed, Solidity-native testing environment, and powerful security features like built-in fuzz testing.[1] Hardhat will be used for more complex integration testing, deployment scripting, and managing interactions with off-chain systems, capitalizing on its powerful JavaScript/TypeScript environment and extensive plugin ecosystem.[1]
- **Account Abstraction Infrastructure: Alchemy Account Kit:** To accelerate development and reduce operational overhead, the MVP will leverage a third-party Infrastructure-as-a-Service (IaaS) provider for its ERC-4337 components. This is a strategic "buy vs. build" decision. Running production-grade Bundler and Paymaster infrastructure is a significant engineering challenge. By using a service like Alchemy's Account Kit, the Tachi team can offload this complexity and focus its limited resources on building the core protocol logic and executing the go-to-market strategy.[1] Alchemy's vertically integrated toolkit provides a production-ready Bundler API, a Gas Manager API for programmable gas sponsorship, and a secure, gas-optimized smart account implementation, directly addressing the needs of the AI crawler client.[1]

The following table provides a consolidated view of the complete technology stack for the Tachi MVP.

| Category | Technology/Service | Version/Standard | Justification |
|---|---|---|---|
| **Publisher Front-End** | Next.js | 14+ (App Router) | Optimal blend of server/client rendering, fast DX, and easy deployment.[15] |
| | Tailwind CSS | 3.x | Utility-first framework for rapid, customizable UI development.[17] |
| | Shadcn/UI | Latest | Provides full code ownership, seamless Tailwind integration, and high-quality accessible |

| | | | |
|---|---|---|---|
| | | | components.[20] |
| **Web3 Integration** | RainbowKit | Latest | Best-in-class "Connect Wallet" UX, abstracting away connection and chain management complexity.[22] |
| | Wagmi | v2 | Comprehensive React Hooks for clean, declarative blockchain interaction.[25] |
| | Viem | v2 | Modern, performant, and type-safe TypeScript interface for Ethereum; the foundation for Wagmi.[1] |
| **Smart Contracts** | Solidity | ^0.8.20 | Most mature and widely supported language for EVM smart contracts.[1] |
| | Foundry | Latest | Primary tool for fast, secure contract development and Solidity-native unit testing/fuzzing.[1] |
| | Hardhat | Latest | Used for complex integration testing and deployment scripting.[1] |
| **Backend/Gateway** | Cloudflare Workers | Latest | Serverless platform for low-latency, global deployment of the publisher gateway logic.[1] |
| **Infrastructure** | Base | L2 Network | Ultra-low transaction fees, EVM compatibility, and deep ecosystem support for USDC.[1] |
| | Alchemy Account Kit | Latest | Production-ready IaaS for ERC-4337 Bundler and Paymaster, accelerating |

|  |  |  |  |
|---|---|---|---|
|  |  |  | time-to-market.[1] |

## Section 4: Building the Publisher Onboarding MVP (Front-End)

This section provides a granular, component-by-component blueprint for constructing the publisher-facing MVP application. This application is the primary tool for demonstrating Tachi's value proposition to potential investors and publisher partners.

### 4.1 Project Setup and Initialization

The foundation of the front-end application will be established through a series of standardized setup commands.

1. **Scaffold the Next.js Project:** A new Next.js project will be created using the create-next-app CLI. The prompts will be configured to include TypeScript and Tailwind CSS integration from the outset, establishing the core project structure.[15] The command pnpm create next-app@latest tachi-mvp --typescript --tailwind --eslint will be used.
2. **Initialize Shadcn/UI:** The Shadcn/UI CLI will be run with the command npx shadcn-ui@latest init. This command will prompt for configuration choices (e.g., style, base color) and then automatically set up the necessary files and dependencies, including globals.css with theme variables, a lib/utils.ts file for helper functions (like cn for merging Tailwind classes), and a components.json file to track installed components.[20]
3. **Install Web3 Dependencies:** All required Web3 libraries will be installed using the package manager. This includes wagmi, viem, @rainbow-me/rainbowkit, and @tanstack/react-query, which is a peer dependency for Wagmi's caching and state management capabilities.[24]
4. **Configure Web3 Providers:** A crucial step is to wrap the entire application in the necessary context providers. A new client component, src/app/providers.tsx, will be created. This component will import WagmiProvider, QueryClientProvider, and RainbowKitProvider and wrap its children prop with them. This is also where the wagmi configuration object will be created using createConfig from Wagmi and getDefaultConfig from RainbowKit. This configuration will specify the target chains (e.g., base) and the application's projectId from WalletConnect Cloud.[24] This Providers component will then be used to wrap the root layout.tsx, making the Web3 context available throughout the entire application.

### 4.2 Core UI Components

The user interface will be composed of several key components, built using the primitives provided by Shadcn/UI.

- **Header Component:** This will be a simple server component serving as the main navigation bar. It will contain the Tachi logo and will conditionally render the ConnectWalletButton client component, ensuring a clean separation of server-rendered layout and client-side interactive elements.
- **ConnectWalletButton Component:** This client component will be a lightweight wrapper around RainbowKit's <ConnectButton /> component.[24] It will allow for custom styling to ensure the button's appearance aligns with the Tachi brand identity. This component is the primary entry point for all user interactions with the protocol.
- **OnboardingWizard Component:** This will be the centerpiece of the MVP. It will be a multi-step form that guides the publisher through the process of creating their on-chain license. It will be built using a combination of Shadcn/UI's Card component for the container and either Tabs or a custom stepper implementation for navigating between steps. The form's state will be managed efficiently using the react-hook-form library, with schema-based validation provided by zod. The wizard will consist of the following steps:
    - **Step 1: Site Information:** A form containing Input and Label components from Shadcn/UI for the publisher to enter their website's name and URL.
    - **Step 2: Set Pricing:** A simple Input field allowing the publisher to define a single, site-wide price-per-crawl in USDC (e.g., a string like "0.005").
    - **Step 3: Create License:** This step will display the protocol's master Terms of Service (which will be fetched from a decentralized storage location like IPFS). The primary action on this screen will be a "Create License NFT" button. Pressing this button will initiate the on-chain transaction to mint the publisher's unique CrawlNFT.
    - **Step 4: Deploy Gateway:** Upon successful creation of the license, this final step will provide the publisher with the pre-configured Cloudflare Worker script needed to enforce the paywall. It will feature a "Copy to Clipboard" Button and will use Shadcn's Toast component to provide immediate visual feedback that the script has been copied successfully.

### 4.3 Web3 Logic and Interaction

The interactive Web3 functionality of the MVP will be powered by Wagmi hooks.

- **Minting the CrawlNFT:** The "Create License NFT" button in the OnboardingWizard will trigger a custom hook, useMintCrawlNFT. This hook will encapsulate the logic for the minting transaction. Internally, it will use Wagmi's useWriteContract hook to prepare and send the transaction to the mint function on the deployed CrawlNFT smart contract.[26] The hook will expose the transaction's status (
isLoading, isSuccess, isError), allowing the UI to react accordingly by showing loading spinners, success messages, or error alerts.
- **Displaying On-Chain Data:** Once the minting transaction is successful, the application

will transition to a dashboard view. This view will use Wagmi's useReadContract hook to fetch data directly from the blockchain and confirm the license's existence to the user. It will call functions like tokenURI and ownerOf on the CrawlNFT contract to display the license details and prove ownership.[26]

- **The Dashboard View:** The initial dashboard will be simple, designed to show two key pieces of information: the publisher's total earnings (which will be $0 initially) and a table of recent, successful crawl events. This table will be populated by querying the blockchain for CrawlLogged events emitted by the ProofOfCrawlLedger contract. To avoid making direct RPC calls from the client, a Next.js API route (/api/crawls) will be created. This server-side route will use a Viem publicClient to query the blockchain for the latest events and return them as a JSON payload, which the front-end can then fetch and display in a table built with Shadcn/UI's Table component.[34]

The following table provides a clear map of the MVP's components, their functions, and the underlying technologies used, serving as a valuable reference for development.

| Component Name | Shadcn/UI Primitives Used | Core Functionality | Key Wagmi/RainbowKit Hook(s) | State Management |
|---|---|---|---|---|
| **ConnectWalletButton** | Button, Dialog | Initiates wallet connection flow and displays connected user state. | <ConnectButton />, useAccount | Handles isConnected, address, chain state provided by RainbowKit. |
| **OnboardingWizard** | Card, Tabs, Input, Label | Guides the user through the multi-step site and license setup process. | N/A (UI component) | Manages form state for site info and pricing using react-hook-form and zod. |
| **MintButton** | Button, Toast | Triggers the on-chain transaction to mint the CrawlNFT license. | useWriteContract | Tracks minting transaction status: isLoading, isSuccess, isError. |
| **DashboardView** | Card, Table | Displays on-chain license data and a log of recent crawl events. | useReadContract | Fetches and displays static on-chain data and dynamic event logs via an API route. |

## Section 5: Core Protocol and Backend Implementation (MVP)

This section provides a condensed but actionable guide for building the smart contract and server-side components of the MVP, drawing directly from the detailed specifications in the provided technical plan.[1]

## 5.1 Smart Contract Development & Deployment

The on-chain foundation of the protocol will be built with a focus on security through simplicity for the MVP.

- **Project Setup:** The development environment will be a hybrid of Hardhat and Foundry. The project will be initialized first with npx hardhat init (selecting the TypeScript option) and then augmented with npx hardhat init-foundry. This setup allows developers to leverage the best features of both toolchains.[1]
- **CrawlNFT.sol:** This contract will serve as the on-chain anchor for each publisher's license. It will be implemented as a standard ERC-721 token, inheriting from OpenZeppelin's battle-tested ERC721.sol and Ownable.sol contracts for access control.[1] For the MVP, the
mint function will be restricted with the onlyOwner modifier, meaning it can only be called by the contract deployer (or a trusted backend service controlled by the Tachi team). Its most critical function is to immutably store the termsOfServiceURI—an IPFS hash pointing to the governing legal terms—which is set at the time of minting.[1] To reinforce its role as a non-tradable credential, the
_beforeTokenTransfer hook will be overridden to revert any transfer attempts after the initial mint.
- **PaymentProcessor.sol & ProofOfCrawlLedger.sol:** As specified in the technical plan, these contracts will be intentionally simple and non-upgradable for the MVP to minimize the security attack surface.[1] The
PaymentProcessor is a stateless contract designed solely to receive USDC payments and forward them to the publisher. The ProofOfCrawlLedger is an append-only log that records successful crawl events, providing the transparent audit trail.
- **Testing:** A rigorous testing strategy is paramount. Unit tests for all smart contract functions and their edge cases will be written in Solidity and executed using Foundry's fast and powerful test runner (forge test).[1] Integration tests, which verify the interactions between the different smart contracts, will be written in TypeScript and run using the Hardhat network, allowing for more complex setup and assertion logic.[1]
- **Deployment:** A deployment script will be written using Hardhat's scripting capabilities. This script will handle the deployment of all three contracts to the Base Goerli testnet. The deployment will be executed via a secure RPC endpoint from a provider like Alchemy or QuickNode.[37] After successful deployment, the contracts' source code will be verified on the Basescan block explorer, promoting transparency and allowing anyone to inspect the on-chain logic.

## 5.2 Publisher Gateway (Cloudflare Worker)

The Publisher Gateway is the off-chain enforcement arm of the protocol, implemented as a serverless function deployed to the edge.

- **Implementation:** The gateway will be developed as a Cloudflare Worker using TypeScript, as detailed in Section 4 of the technical plan.[1] This choice leverages Cloudflare's global network for low-latency request interception.[39]
- **Key Logic Flow:**
  1. **Request Interception & Identification:** The worker intercepts every incoming request to the publisher's site. For the MVP, it will identify known AI crawlers by matching the User-Agent header against a maintained list.[1]
  2. **Issue 402 Challenge:** If a request is identified as a target crawler and does not contain a valid payment proof in the Authorization header, the worker will halt the request and return an HTTP 402 Payment Required response. This response will include the custom x402-* headers specifying the price, currency (USDC), recipient (the PaymentProcessor contract address), and network (Base).[1]
  3. **Verify Payment Proof:** When the crawler makes a subsequent request including an Authorization: Bearer <tx_hash> header, the worker must verify this proof. It will use a Viem publicClient to make a JSON-RPC call to a Base network endpoint.[1] It will call publicClient.getTransactionReceipt({ hash: tx_hash }) to fetch the transaction details.[1] The worker will then parse the receipt's logs to validate that a USDC transfer was successfully executed, to the correct PaymentProcessor contract, for an amount greater than or equal to the required price.
  4. **Deliver Content & Log Crawl:** If verification is successful, the worker will allow the request to proceed to the origin server to fetch the actual content, which is then returned to the crawler. To avoid adding latency to the content delivery, the worker will use the ctx.waitUntil() method to asynchronously trigger a final on-chain transaction: a call to the logCrawl function on the ProofOfCrawlLedger contract, immutably recording the successful interaction.[1]

## 5.3 AI Crawler SDK (Proof of Concept)

To facilitate end-to-end testing and provide a reference implementation for future AI partners, a proof-of-concept (PoC) AI Crawler SDK will be developed.

- **Implementation:** For the MVP, this will be a simple Node.js package written in TypeScript.[1]
- Key Logic Flow (as detailed in Section 5 of the technical plan [1]):

1. The SDK will expose a high-level wrapper function, such as fetchWithTachi(url).
2. This function will initially make a standard fetch request to the target URL.
3. If it receives a 402 response, it will automatically parse the x402-* headers to extract the payment details.
4. It will then use Alchemy's aa-sdk to programmatically construct a UserOperation object. This operation will encode a call for the crawler's ERC-4337 smart wallet to execute the required USDC transfer.
5. The SDK will send this signed UserOperation to Alchemy's Bundler API endpoint for execution on the Base network.
6. After polling for and receiving confirmation that the payment transaction has been successfully mined, the SDK will automatically make a second request to the original URL, this time including the Authorization: Bearer <tx_hash> header.
7. Finally, it will return the content from the successful second request to the original caller, abstracting away the entire payment negotiation process.

## Section 6: MVP Integration and End-to-End Testing

The final phase of MVP development involves integrating all components and conducting a thorough end-to-end test in a controlled environment to validate the entire protocol flow.

### 6.1 Environment Setup

A dedicated testing environment will be established, comprising the following elements:

- **On-Chain Contracts:** The three core smart contracts (CrawlNFT, PaymentProcessor, ProofOfCrawlLedger) will be deployed to the Base Goerli testnet. Their addresses will be recorded and used to configure the other components.
- **Front-End Application:** The Next.js publisher MVP will be deployed to Vercel's preview environment. It will be configured with the necessary environment variables, including the deployed contract addresses and a Base Goerli RPC URL.
- **Publisher Test Site:** A simple, static website (e.g., a few HTML pages deployed via Cloudflare Pages) will be created to serve as the target for the AI crawler.

### 6.2 The "Happy Path" Test Scenario

A comprehensive end-to-end test will be executed to simulate the complete lifecycle of a single pay-per-crawl transaction.

1. **Publisher Onboarding:** Use the deployed Tachi MVP front-end. Connect a wallet (funded with Base Goerli ETH) and proceed through the OnboardingWizard to mint a new CrawlNFT for the publisher test site.
2. **Gateway Deployment:** Copy the Cloudflare Worker script generated by the MVP in the

final onboarding step. Deploy this worker script to the Cloudflare account associated with the publisher test site.

3. **Crawler Setup:** Using the PoC SDK and Alchemy's tools, create and provision a new ERC-4337 smart wallet. Fund this wallet with a small amount of Base Goerli USDC, which can be obtained from a faucet or swapped for.
4. **Execute Crawl:** Run the PoC AI Crawler SDK script, instructing it to fetch a URL from the publisher test site.
5. **Verification:** The success of the test will be verified by checking multiple sources:
    - **Crawler Output:** The crawler script's console output should confirm that it successfully received the content of the target page.
    - **Block Explorer (Basescan):** The Base Goerli block explorer will be used to verify that two key on-chain events occurred: a USDC Transfer event from the crawler's smart wallet to the PaymentProcessor contract, and a CrawlLogged event emitted by the ProofOfCrawlLedger contract.
    - **Tachi Dashboard:** The publisher MVP dashboard should be refreshed, and the "Recent Crawls" table should now display a new entry corresponding to the test crawl, confirming that the front-end is correctly reading and displaying the on-chain log.

The following table outlines a tangible, week-by-week sprint plan for building and testing the MVP within a 90-day timeframe.

| Sprint (2 Weeks) | Name & Goal | Key Tasks |
|---|---|---|
| **1 (Weeks 1-2)** | Foundation & Setup<br>Goal: All project scaffolding and Web3 provider configurations are complete. | - Initialize Next.js project with TypeScript, Tailwind CSS.<br>- Initialize Shadcn/UI.<br>- Install all Web3 dependencies.<br>- Create and configure WagmiProvider, QueryClientProvider, and RainbowKitProvider.<br>- Set up hybrid Foundry/Hardhat smart contract repository. |
| **2 (Weeks 3-4)** | Smart Contract Implementation<br>Goal: Core MVP smart contracts are written, tested, and ready for deployment. | - Implement CrawlNFT.sol, PaymentProcessor.sol, and ProofOfCrawlLedger.sol.<br>- Write comprehensive unit tests in Solidity using Foundry.<br>- Write basic integration tests in TypeScript using Hardhat. |
| **3 (Weeks 5-6)** | Front-End Onboarding UI<br>Goal: A publisher can connect | - Build the Header and ConnectWalletButton |

|  |  |  |
|---|---|---|
|  | their wallet and complete the initial onboarding steps. | components.<br>- Build the UI for the OnboardingWizard (Steps 1 & 2) using Shadcn/UI components.<br>- Integrate react-hook-form and zod for form state management and validation. |
| **4 (Weeks 7-8)** | On-Chain Front-End Logic<br>Goal: A publisher can successfully mint their license NFT from the UI. | - Implement the useMintCrawlNFT custom hook using Wagmi's useWriteContract.<br>- Connect the minting logic to the "Create License NFT" button in Step 3 of the wizard.<br>- Deploy all smart contracts to the Base Goerli testnet and verify on Basescan. |
| **5 (Weeks 9-10)** | Publisher Gateway & Dashboard<br>Goal: The gateway logic is complete and the dashboard displays on-chain data. | - Implement the full logic for the Cloudflare Worker gateway in TypeScript.<br>- Build the DashboardView UI, including the earnings display and recent crawls table.<br>- Create the Next.js API route (/api/crawls) to fetch and return crawl logs from the blockchain. |
| **6 (Weeks 11-12)** | E2E Integration & Polish<br>Goal: A polished, fully functional end-to-end demo is ready for presentation. | - Build the PoC AI Crawler SDK in Node.js.<br>- Conduct the full "Happy Path" end-to-end integration test.<br>- Refine UI/UX, fix any outstanding bugs, and prepare presentation materials. |

## Part III: Beyond the MVP: Scaling and Recommendations

This final part of the report provides a strategic outlook beyond the initial Minimum Viable Product. It outlines the phased evolution from a validated MVP into a mature, decentralized protocol and concludes with a set of actionable recommendations for the founder to guide the venture's critical early stages.

## Section 7: The Roadmap to Liquid Data Markets and Decentralization

The MVP is designed as a focused starting point, not the final destination. The long-term vision for the Tachi protocol is to evolve from a functional but centralized product into a fully decentralized, community-owned piece of public infrastructure for the AI-driven web. This roadmap outlines a deliberate, three-phased journey to achieve that vision, systematically addressing product, market, and governance risks in a logical sequence.[1]

### 7.1 Phase 2: V1.0 - From Payments to Markets (Months 7-12)

Upon successful validation of the MVP with the "atomic network," the next phase focuses on introducing liquidity and sophisticated pricing mechanisms, transforming data access rights into a tradable, liquid asset class. This evolution represents a key strategic differentiator that centralized, fiat-based competitors cannot easily replicate.[1]
The core technical milestone of this phase is the introduction of a new smart contract, CrawlToken.sol, an ERC-20 token representing fungible, pre-paid rights to access a specific publisher's content.[1] Each publisher's
CrawlNFT will govern its own unique CrawlToken. A corresponding CrawlTokenMinter.sol contract will allow the publisher (the owner of the CrawlNFT) to mint new batches of their tokens—for example, minting 1,000,000 tokens, each redeemable for a single crawl on their site.
The protocol's core contracts will then be upgraded (using a secure proxy pattern like UUPS) to accept these CrawlTokens as a payment method alongside USDC.[1] This gives AI companies a new, more flexible way to procure data access. They can continue to pay per-request with stablecoins, or they can redeem a pre-purchased
CrawlToken.
The most significant strategic element of this phase is actively facilitating the creation of liquidity pools for these new CrawlToken/USDC pairs on Base-native decentralized exchanges (DEXs) like Uniswap. This crucial step transforms illiquid, bespoke data licenses into vibrant, liquid assets. It enables dynamic, market-driven price discovery for data access rights, allowing AI companies to buy tokens on the open market and even allowing for speculation and hedging by data-focused traders.[1] This creates a true, open market for data access—a powerful primitive for the new web economy.

### 7.2 Phase 3: V2.0 - The Protocol as a Public Good (Months 13+)

The final phase of the roadmap completes the transition from a company-controlled product to a credibly neutral, community-owned public utility. This ensures the protocol's long-term resilience and aligns it with the ethos of the open web, solidifying its position as a trusted standard.[1]
The central event of this phase is the launch of a native governance token and the establishment of a Protocol DAO (Decentralized Autonomous Organization).[1] The token's distribution model will be strategically designed to empower all key stakeholders. A significant portion of the token supply should be allocated for a retroactive airdrop to early-adopting publishers and AI crawlers, rewarding their participation and decentralizing governance from day one.[1] Further allocations will be made to a community-governed treasury and to the founding team and investors (subject to a vesting schedule).
The DAO itself will be established using a battle-tested on-chain governance framework, such as OpenZeppelin's implementation of the Governor standard. This DAO will have control over a community treasury, funded by a small, protocol-wide fee on all transactions. It will be empowered to make binding on-chain decisions, such as upgrading protocol smart contracts, allocating treasury funds for ecosystem grants, or adjusting protocol-level fees.[1]
The culmination of this phase is the "progressive decentralization" of the protocol. The administrative privileges and ownership of the core smart contracts will be formally transferred from the founding team's multi-signature wallet to the DAO's governance contract. At this point, the protocol achieves credible neutrality, operating as a self-sustaining public good governed by its community of stakeholders.[1]

## Section 8: Concluding Recommendations for the Founder

This report provides a comprehensive strategic and technical playbook for launching the Tachi protocol. Success will depend on relentless execution, strategic focus, and a deep understanding of the market dynamics at play. The following are the most critical concluding recommendations.

### 8.1 Immediate Priorities (First 90 Days)

The next three months are critical for building momentum and de-risking the venture. The founder's focus should be absolute and directed toward three parallel objectives:

1. **Build the MVP:** Execute the 6-sprint development plan outlined in Section 6.2 with relentless focus. The primary goal is to produce a polished, functional end-to-end demo that can be put in front of potential investors and publisher partners.
2. **Engage Legal Counsel:** Do not delay on the legal front. Immediately engage qualified

legal counsel in the US and EU to begin the process of drafting the master terms of service and, most importantly, commissioning the formal legal opinion letters on the enforceability of the on-chain licensing model. These documents are critical assets for the business development process.
3. **Target the Atomic Network:** The GTM effort must begin now. Start building relationships and initiating conversations with the identified list of high-value financial and technology publications. The goal is to have a "coalition of the willing" ready to pilot the protocol as soon as the MVP is complete.

## 8.2 Key Hiring Considerations

While the founder is technically proficient, building a protocol and a company requires a team with a diverse skill set. The research highlights potential talent gaps that should be addressed with early, strategic hires.[1]

- **First Hire: Head of Publisher Partnerships / Business Development:** The single greatest risk to the venture is the "cold start" problem. The first key hire should be an individual with a deep network and extensive experience within the digital media and publishing industries. This person's primary responsibility will be to build the publisher coalition, navigate the complex organizational structures of media companies, and secure the initial partnerships that will form the "atomic network."
- **Second Hire: Senior Protocol Engineer:** To augment the founding team's capabilities and ensure the security and scalability of the on-chain components, the second key hire should be a senior engineer with deep, demonstrated expertise in smart contract security, L2 architecture, and protocol design. This individual will be critical for conducting rigorous internal code reviews, managing external security audits, and architecting the V1.0 and V2.0 upgrades.

## 8.3 Final Strategic Checklist

To maintain focus on the factors most critical to success, the founder should continuously evaluate progress against the following strategic questions:

- **Publisher Friction:** At every stage of product development and onboarding, are we ruthlessly minimizing friction for the publisher? Is the process as simple, intuitive, and secure as it can possibly be?
- **Exclusive Content Mass:** How can we accelerate the path to achieving a critical mass of exclusive, high-value, non-commoditized content on the protocol? Is our "atomic network" strategy delivering the leverage we need?
- **Value Proposition Clarity:** Is our messaging to both publishers and AI companies clear, compelling, and differentiated? Are we effectively communicating the unique value of an open, decentralized standard versus a closed, proprietary service?
- **Capital Efficiency:** Are we deploying capital in a way that maximally de-risks the

venture and moves us closer to key milestones? Are we making the right "build vs. buy" decisions to conserve resources and accelerate time-to-market?

By adhering to the detailed technical plan in this report and maintaining a sharp focus on these strategic imperatives, the Tachi protocol is well-positioned to not only succeed as a venture but to play a pivotal role in defining a more equitable and sustainable economic model for the AI-driven web.

## Works cited

1. Pay-Per-Crawl Startup Technical Plan
2. accessed December 31, 1969, uploaded:Pay-Per-Crawl Startup Technical Plan
3. 23 Best Finance Newsletters To Keep You Updated in 2025 - The CFO Club, accessed July 16, 2025, https://thecfoclub.com/career/best-finance-newsletters/
4. Best Financial Newsletters - Career Development Yale School of Management, accessed July 16, 2025, https://cdo.som.yale.edu/blog/2025/06/13/best-financial-newsletters/
5. Top 10 Investment Newsletters for Inspiration - Flodesk, accessed July 16, 2025, https://flodesk.com/tips/investment-newsletters
6. Forbes Newsletters, accessed July 16, 2025, https://account.forbes.com/newsletters/
7. Financial Newsletter: Investing & Markets - Morgan Stanley, accessed July 16, 2025, https://www.morganstanley.com/newsletter/subscribe
8. 18 Best Tech Blogs & News Sites For Career Growth | CaffeinatedKyle.com, accessed July 16, 2025, https://caffeinatedkyle.com/best-tech-blogs/
9. The 50 Best Tech Blogs, Ranked by Popularity - Detailed.com, accessed July 16, 2025, https://detailed.com/tech-blogs/
10. List of Open Access Journals, accessed July 16, 2025, https://sec.edu.in/feed/openaccessjournals28052020.pdf
11. List of open-access journals - Wikipedia, accessed July 16, 2025, https://en.wikipedia.org/wiki/List_of_open-access_journals
12. Journals - Diamond Scientific Publishing Open Access Journals, accessed July 16, 2025, https://www.diamondopen.com/journals-list/
13. Alphabetical List of Open Access Journals - Open Access Journal Resources - LibGuides, accessed July 16, 2025, https://cccs.libguides.com/c.php?g=1311837&p=9641895
14. Open Access Journals - Elsevier, accessed July 16, 2025, https://www.elsevier.com/open-access/open-access-journals
15. Next.js documentation - DevDocs, accessed July 16, 2025, https://devdocs.io/nextjs/
16. Next.js on Vercel, accessed July 16, 2025, https://vercel.com/docs/frameworks/nextjs
17. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML., accessed July 16, 2025, https://tailwindcss.com/
18. Theme variables - Core concepts - Tailwind CSS, accessed July 16, 2025, https://tailwindcss.com/docs/theme

19. Functions and directives - Core concepts - Tailwind CSS, accessed July 16, 2025, https://tailwindcss.com/docs/functions-and-directives
20. Shadcn UI for Beginners: The Ultimate Step-by-Step Tutorial - CodeParrot, accessed July 16, 2025, https://codeparrot.ai/blogs/shadcn-ui-for-beginners-the-ultimate-guide-and-step-by-step-tutorial
21. Introduction - shadcn/ui, accessed July 16, 2025, https://ui.shadcn.com/docs
22. Introduction - RainbowKit, accessed July 16, 2025, https://rainbowkit.com/docs/introduction
23. RainbowKit, accessed July 16, 2025, https://rainbowkit.com/
24. Installation — RainbowKit, accessed July 16, 2025, https://v1.rainbowkit.com/docs/installation
25. Wagmi | Reactivity for Ethereum apps, accessed July 16, 2025, https://wagmi.sh/
26. Getting Started | Wagmi, accessed July 16, 2025, https://wagmi.sh/react/getting-started
27. Getting Started · Viem, accessed July 16, 2025, https://viem.sh/docs/getting-started.html
28. Hardhat vs Foundry: Choosing the Right Ethereum Development Tool - MetaMask, accessed July 16, 2025, https://metamask.io/news/hardhat-vs-foundry-choosing-the-right-ethereum-development-tool
29. Integrating with Foundry | Ethereum development environment for ..., accessed July 16, 2025, https://hardhat.org/hardhat-runner/docs/advanced/hardhat-and-foundry
30. Introducing Account Kit - Alchemy, accessed July 16, 2025, https://www.alchemy.com/blog/introducing-account-kit
31. ERC-4337 Bundler API for Account Abstraction - Alchemy, accessed July 16, 2025, https://www.alchemy.com/bundler
32. Framework guides - Tailwind CSS, accessed July 16, 2025, https://tailwindcss.com/docs/installation/framework-guides
33. Installation - RainbowKit, accessed July 16, 2025, https://rainbowkit.com/docs/installation
34. Public Client - Viem, accessed July 16, 2025, https://viem.sh/docs/clients/public
35. ERC721 - OpenZeppelin Docs, accessed July 16, 2025, https://docs.openzeppelin.com/contracts/3.x/erc721
36. How to Create ERC-721 NFTs on Ethereum with OpenZeppelin: A Step-by-Step Tutorial, accessed July 16, 2025, https://medium.com/coinmonks/how-to-create-erc-721-nfts-on-ethereum-with-openzeppelin-a-step-by-step-tutorial-47b252843dd9
37. Ultimate Guide to Deploying Smart Contracts on Base 2024 - Rapid Innovation, accessed July 16, 2025, https://www.rapidinnovation.io/post/deploy-a-smart-contract-on-base
38. How to Deploy a Smart Contract on Base - thirdweb blog, accessed July 16, 2025, https://blog.thirdweb.com/guides/how-to-deploy-a-smart-contract-to-base-network-testnet-coinbase-l2/

39. Development & testing · Cloudflare Workers docs, accessed July 16, 2025, https://developers.cloudflare.com/workers/development-testing/