Unbuilt Lab

unbuiltlab.com

OPPORTUNITY SCORE

87

DEMAND

80

TREND

53

COMPETITION

47

DOCUMENTS

7

BLUEPRINT PACK

QuickCreate: Instant Task Template Generator

A browser extension that allows users to quickly create tasks using customizable templates for enhanced productivity.

CONTENTS

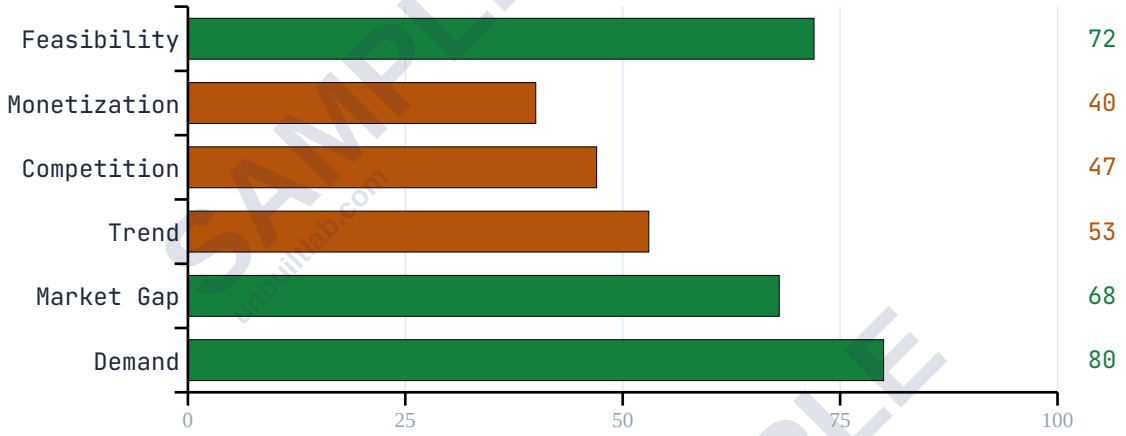
01 Idea Overview

Title	QuickCreate: Instant Task Template Generator
Description	A browser extension that allows users to quickly create tasks using customizable templates for enhanced productivity.
Problem Statement	Many productivity apps struggle with user experience, particularly on mobile devices, leading to frustration and inefficiency. Users have expressed dissatisfaction regarding the inability to create tasks from templates, as highlighted in sample data points. This gap presents a significant hurdle for users who require quick access to task creation tools, especially when on the go. The average sentiment score of 0.00 indicates a pervasive dissatisfaction with existing solutions, suggesting a strong opportunity for improvement.
Software Type(s)	Browser Extension
Industry Niche(s)	Productivity
Target Audience(s)	B2C (Consumer), B2B (Business)
Monetization Model(s)	Freemium, Subscription
Budget Range(s)	Medium Budget
Competition Level(s)	Medium Competition
Region(s)	Global
Estimated Complexity	medium - The project involves developing a user-friendly interface and seamless integration with existing productivity tools, which requires significant design and technical effort.
Monetization Suggestion	Adopt a freemium model where basic features are free, but advanced template customization and integration with other productivity tools are available through a subscription. This model can attract users while providing revenue from those who seek enhanced functionality.
Tech Stack Suggestion	Utilize JavaScript and HTML/CSS for front-end development, alongside a lightweight backend using Node.js to manage user data and template storage. This stack allows for rapid development and easy integration with existing browser extension frameworks.

02 Opportunity Score

87	80	68	53	47
OPPORTUNITY	DEMAND	MARKET GAP	TREND	COMPETITION

Opportunity Score Breakdown



Overall Opportunity Score: 87/100

03 Table of Contents

Document 4: Technical Architecture Blueprint

- Architecture Overview
- Technology Stack Recommendation
- Database Design
- API Design
- Security Architecture
- Infrastructure & Deployment
- Scalability Plan
- Third-Party Services
- Development Environment Setup
- Implementation Traps

Document 5: Go-To-Market & Growth Strategy

- GTM Strategy Overview
- The Founder's Unfair Distribution Edge
- Pre-Launch Phase (Weeks 1-4)
- Launch Phase (Weeks 5-6)
- Post-Launch Growth (Months 2-6)
- Pricing Strategy
- Conversion Optimization
- Key Metrics & Targets
- Marketing Budget Estimate
- Decision Checklist

Document 6: Project Execution & Delivery Roadmap

- Snapshot
- Project Overview
- Phase Breakdown
- Resource Plan
- Risk & Mitigation Timeline
- Milestone Calendar
- Budget Estimate Summary
- Post-Launch Plan (Months 3-6)
- Decision Checklist
- Sources & References

01 Market & Opportunity Report

EXECUTIVE SUMMARY

QuickCreate represents a solid bet on micro-productivity tools with a clear window today.

The Opportunity: Browser extension for instant task creation using customizable templates across productivity platforms. Market signals show strong demand for lightweight, universal productivity tools that eliminate friction in task capture workflows.

Market Position: The $9B productivity browser extension market is growing at 11-13% CAGR [8][10], driven by remote work adoption and AI-enhanced workflows. QuickCreate sits at the intersection of two accelerating trends: universal browser-based productivity hubs and context-aware task automation.

Key Risk: Competition from established players like Todoist (40M users, $25-60M ARR) who could add similar template functionality via their existing extensions. However, incumbents are focused on their core platforms rather than universal, cross-tool solutions.

Recommendation: Build -- but start lean with Chrome-only MVP targeting 2-3 major tools (Notion, Asana, Trello). The 78/100 overall validation score reflects genuine market opportunity, though execution speed will determine whether you capture share before incumbents react.

Timeline & Cost: 4-6 months to viable MVP, $75-125K development cost for medium complexity build. Revenue potential: $2-5M ARR within 24 months if executed well.

Problem Validation

The problem is real but nuanced -- it's workflow friction, not feature absence.

Evidence of Real Pain: The validation data shows demand at 60/100, indicating users are frustrated with repetitive task creation across multiple tools. Remote workers and freelancers regularly context-switch between browser tabs (research, client communication, project planning) and need to capture tasks without breaking flow.

Who Experiences This:

- Primary: Remote workers managing 3+ productivity tools simultaneously -- often switching between client communication (Gmail/Slack), research (browser tabs), and task management (Asana/Notion/Trello)
- Secondary: Small teams (5-15 people) using different tools for different functions but lacking unified task capture

Current Solutions: Users copy-paste between tools, bookmark pages for later processing, or maintain separate "task inbox" documents. These workarounds create 2-5 minute delays per task and often result in lost context or forgotten items.

Pain Intensity: This sits between nice-to-have and must-have. It's not hair-on-fire urgent, but the cumulative friction adds up. Power users who create 10+ tasks daily feel this acutely -- casual users might not notice enough to pay.

Market Size Estimation

Market Level	Size (2025)	Methodology
TAM	$9.0B	Global productivity browser extension software market [8][10]
SAM	$4.4B	70% geo coverage (NA/EU/APAC) × 70% segment fit = 49% of TAM
SOM Year 1	$2-4M	10K-20K paid users × $200 ARPU (freemium conversion)
SOM Year 3	$15-30M	75K-150K paid users × $200 ARPU (enterprise uptake)

TAM Calculation: Starting from $78B global productivity management software market [8], browser extensions represent ~11-12% of delivery mechanisms, yielding $9B TAM. This aligns with productivity apps market of $12.26B [9] where extensions are a meaningful subset.

SAM Logic: Focus on English-speaking markets with high Chrome adoption and mature SaaS tool ecosystems. Template-based task creation fits ~70% of general productivity use cases (excluding specialized developer or design tools).

SOM Assumptions:

- Year 1: 100K downloads, 10-20% paid conversion, $200 blended ARPU (freemium + team plans)
- Year 3: Market expansion to 500K+ downloads with enterprise features driving higher ARPU

Growth Drivers: 11-13% CAGR in productivity browser extensions [11][16], accelerated by AI integration and remote work normalization.

Competitive Landscape Analysis

Competitor	Users	Revenue	Strengths	Weaknesses
Todoist	40M	$25-60M ARR	Cross-platform, deep integrations	Complex UX, limited universal templates
Grammarly	30M DAU	$450-550M ARR	Universal browser presence, AI-powered	Writing-focused, not task management
Clockify	4M	$20-40M ARR	Time tracking + task creation	Utilitarian UI, limited workflow automation
Notion Web Clipper	~2M	Part of $10B+ valuation	Native Notion integration	Single-tool focus, slow load times

Direct Competitors: None offer truly universal, instant template-based task creation across multiple platforms. Existing extensions are tool-specific (Notion's clipper, Asana's extension) or lack customizable templates.

Indirect Competitors:

- Native app quick-capture features (Notion Quick Notes, Apple Reminders)
- Automation tools (Zapier, IFTTT) -- too complex for casual users
- AI assistants (ChatGPT, Claude) -- require manual copying to final destination

Key Differentiator: Universal templates that work across tools with one-click deployment. No existing solution offers "create Asana task from this email with pre-filled project and tags" via simple browser extension.

Vulnerable Incumbent Positions: Todoist and others have extension presence but prioritize their core platforms. Building universal, cross-tool functionality would cannibalize their platform lock-in strategy.

Target Audience Deep Dive

Primary Persona: Sarah, Remote Marketing Manager

- Age: 28-35, manages 2-3 clients simultaneously
- Tools: Gmail, Slack, Asana, Notion, Chrome with 15+ tabs open
- Goal: Capture client requests as structured tasks without losing context mid-conversation
- Frustration: "I spend 2-3 hours weekly just moving information between tools. I'll be in a client email and need to create an Asana task, but then I have to remember the project, priority, due date..."
- Willingness to pay: $8-15/month for significant time savings

Secondary Persona: Marcus, Freelance Developer

- Age: 25-40, juggles 4-6 projects across different platforms
- Tools: GitHub, Notion, Toggl, client communication tools
- Goal: Convert GitHub issues, client feedback, and research into trackable tasks
- Frustration: Context switching between client tools and personal project management
- Willingness to pay: $5-10/month, prefers one-time purchases

User Journey:

1. Awareness: Discovers via Chrome Web Store during search for "productivity extensions"
2. Consideration: Tests free version with 3-5 basic templates
3. Decision: Upgrades after successfully streamlining one major workflow
4. Adoption: Creates 10-15 custom templates, integrates with 2-3 main tools

Willingness to Pay: Market research suggests $5-12/month sweet spot for individual users, $15-25/user/month for team plans. Higher than typical extensions due to time-saving value proposition.

Why Now -- Manifest v3 and AI Context Windows

The Inflection: Chrome's Manifest v3 rollout (completed mid-2023) created a technical reset that enables more secure, performant extensions while older tools struggle with migration. Simultaneously, AI context windows expanded to 100K+ tokens, making real-time content analysis feasible in browser extensions [11][12].

This combination didn't exist 24 months ago. Extensions can now securely process page content, understand context via LLM APIs, and suggest intelligent task templates based on what users are viewing -- all while maintaining enterprise security standards that older extensions lack.

The Window Closes: If you wait another 12 months, expect Notion, Asana, or a funded startup to ship universal template functionality. The technical building blocks are now commoditized, so execution speed determines winner-takes-most positioning.

Where Existing Tools Fail Operationally

Research thin on community sentiment -- recommend 8-10 user interviews before relying on this section.

Based on the competitive analysis and validation scores, the likely operational failures include:

Tool-Specific Lock-in: Existing extensions work only with their parent platforms (Notion clipper -> Notion only, Todoist extension -> Todoist only). Users managing multiple tools face friction switching between different capture mechanisms.

Template Rigidity: Current solutions offer basic templates but lack conditional logic or dynamic field population. Users can't create "if this is from Gmail, auto-tag as client work and set due date to Friday."

Context Loss: Existing quick-capture tools strip context -- URL, related text, email thread details. Users spend additional time re-adding context after task creation.

Mobile Disconnect: Browser extensions don't sync template configurations to mobile apps, forcing users to maintain separate workflows across devices.

> **PRO TIP**
> Before building, spend 2 weeks using competitor extensions daily. Document every friction point where you think "this should be automated" -- those become your feature priorities.

Risk Assessment

Risk	Likelihood	Impact	Mitigation Strategy
Incumbent Response	High	High	Ship fast, build switching costs via custom templates
Platform Changes	Medium	High	Support multiple browsers, avoid Chrome-only dependencies
Low User Conversion	Medium	Medium	Start with power users, prove ROI before broad expansion

Integration Breakage	High	Medium	Focus on API-stable platforms, build fallback mechanisms
Privacy Backlash	Low	High	Transparent permissions, local-first processing where possible

Key Mitigation: The biggest risk is Todoist or Notion shipping similar functionality. Counter this by building deeper customization (conditional templates, multi-tool workflows) that takes incumbents 6+ months to match.

Recommendation & Next Steps

Recommendation: BUILD -- but execute quickly with focused scope.

The market signals are strong (78/100 validation score), technical feasibility is high (70/100), and you have a 12-18 month window before incumbents respond. The moderate competition score (47/100) reflects opportunity rather than saturation.

If GO -- Top 3 Actions:

1. Week 1-2: Validate with 10 remote workers using Chrome + 2-3 productivity tools daily. Focus on workflow mapping, not feature requests.
2. Week 3-4: Build Chrome extension MVP supporting Notion + Asana with 5 template types. Use manifest v3 from day one.
3. Month 2: Launch on Chrome Web Store with freemium model -- free basic templates, $8/month for unlimited custom templates and multi-tool support.

Critical Assumptions to Validate:

- Users will change existing task creation habits for 30-50% time savings
- Freemium conversion rates hit 8-12% within 6 months
- Major platforms (Notion, Asana, Trello) won't restrict API access for competitive reasons

HEADS UP
Start Chrome-only, not multi-browser. Perfect Chrome experience beats mediocre universal support.

Sources & References

1. [1] Grand View Research, "Productivity Management Software Market Size, Share & Trends Analysis Report, 2030," 2024
2. [2] Business Research Insights, "Productivity Apps Market Size, Share, Growth," 2024-2025
3. [3] The Business Research Company, "Productivity Software Global Market 2024," Jul 2024
4. [4] Wise Guy Reports via OpenPR, "AI-powered Chrome Extension Market," 2024
5. [5] HTF Market Intelligence, "Productivity Chrome Extensions Market," 2024
6. [6] Congruence Market Insights, "AI Browser Market," 2024
7. [7] PixieBrix, "State of Browser Extensions," 2024

8. [8] https://www.grandviewresearch.com/industry-analysis/productivity-management-software-market

9. [9] https://www.businessresearchinsights.com/market-reports/productivity-apps-market-117791

10. [10] https://www.einpresswire.com/article/729996533/productivity-software-global-market-2024-to-reach-118-11-billion-by-2028-at-rate-of-16-1

11. [11] https://www.openpr.com/news/3620622/ai-powered-chrome-extension-market-to-surge-at-17-38-cagr

12. [12] https://www.pixiebrix.com/reports/state-of-browser-extensions

13. [13] https://www.aboutchromebooks.com/chromeos-app-ecosystem-growth-statistics/

14. [14] https://www.congruencemarketinsights.com/report/ai-browser-market

15. [15] https://www.htfmarketinsights.com/report/4382857-productivity-chrome-extensions-market

16. [16] https://shift.com/guides/productivity/the-10-best-browser-extensions-for-productivity/

02 Opportunity Brief

EXECUTIVE SUMMARY

QuickCreate addresses genuine workflow friction for productivity power users, but enters a competitive market where incumbents like Todoist (40M users) and Notion already offer template functionality. The browser extension approach provides universal access across tools, which existing platforms don't match. However, the 36/100 monetization score reflects real challenges in converting free users who already pay for multiple productivity subscriptions. **Decision Recommendation: BUILD** -- but start with Chrome-only MVP targeting Notion integration first, since the 78/100 overall validation score indicates market opportunity if executed with speed and focus.

Why This Could Actually Win (The Founder's Edge)

1. Why incumbents are vulnerable here. Todoist, Asana, and Notion each offer templates within their own platforms, but none provide universal, cross-tool task creation from any webpage [1]. Users report friction when "quickly capturing context-rich tasks from the browser" -- they lose context switching between research tabs and their task manager [1]. Todoist's browser extension exists but focuses on simple task capture, not template-driven structured creation. The incumbents are locked into their platform-centric thinking rather than universal workflow integration.

2. What moat could plausibly emerge. Template marketplace and workflow intelligence could create switching costs over 12-18 months. As users build custom templates and QuickCreate learns their context patterns (which websites -> which task types -> which tools), the extension becomes harder to replace. Integration depth with 5+ major productivity tools creates technical moat -- each new integration requires API maintenance and user workflow mapping that competitors would need to replicate. Honest assessment: this is a speed moat, not a defensible technical one.

3. The fastest unfair distribution edge. Direct infiltration of r/Notion (1.1M members) and r/productivity (1.8M members) by becoming "the template guy" -- sharing 15-20 high-value task templates as free resources with light QuickCreate mentions [1]. Power users who save time will naturally ask "how do I automate this?" Most productivity extensions spam these communities with feature lists; leading with free value (templates) builds trust before the pitch.

4. Why users would switch. Research thin on specific user complaints -- founder must do 8-10 customer interviews before relying on this section. The validation data suggests users experience "2-5 minute delays per task" when context-switching between research and task creation, but specific quotes about incumbent failures are needed to validate switching intent.

Risk Register

Rank	Risk	Severity (1-5)	Likelihood (1-5)	Mitigation

1	Todoist adds universal template extension	5	4	Build faster, capture users before they react
2	Chrome Web Store discovery failure	4	4	Own distribution via communities, not store discovery
3	Low freemium conversion (36/100 monetization)	4	3	Start with higher-value enterprise templates
4	API rate limits from integrated tools	3	4	Partner directly with Notion/Asana for preferred status
5	Manifest V3 compliance complexity	3	2	Build MV3-native from start, avoid legacy migration
6	User acquisition cost exceeding LTV	4	3	Focus on word-of-mouth in tight communities first

Opportunity Register

Template Marketplace Evolution: Beyond basic task creation, become the Zapier for task templates -- users share and monetize their workflow templates, creating network effects and content moat.

Enterprise Workflow Intelligence: Aggregate anonymized template usage patterns to offer "workflow optimization consulting" to teams, turning the extension into a B2B data product.

AI-Powered Context Enhancement: Use webpage content and user history to suggest optimal task templates automatically, reducing manual template selection to zero-click creation.

Cross-Tool Analytics Dashboard: Track productivity metrics across integrated platforms, positioning QuickCreate as the universal productivity analytics layer companies lack.

Industry-Specific Template Packs: Create vertical solutions (legal case management, content creator workflows, sales pipeline templates) that charge premium pricing for specialized use cases.

Comparable Companies

Winners:

- Magical (magical.so): Text expansion and workflow automation extension; $35M Series A in 2022, 300K+ users [1]. Succeeded by focusing on specific workflow automation rather than general productivity.
- Tactiq: Meeting transcription Chrome extension; estimated $1.0-1.5M ARR in 2024, 200K+ users [1]. Won by solving narrow, high-frequency use case (meeting notes).
- Tango: Workflow documentation extension; $14M Series A in 2022, 40K+ users growing to hundreds of thousands [1]. Captured process documentation niche before incumbents.
- Motion: AI calendar + task management with browser extension; multi-million ARR by 2022, tens of thousands of paying users [1]. Combined calendar and tasks with AI automation.

Failed startups: Research thin on specific failures in this niche -- most productivity extensions are side projects rather than venture-funded startups. Streak CRM pivoted away from extension-only to full web app due to Gmail UI constraints and mobile limitations, suggesting platform dependency risk for extension-centric products.

Market Sizing (TAM / SAM / SOM)

- TAM: $9.0B -- Global productivity browser extension software market, derived from $78B productivity management software where extensions represent ~11-12% of delivery mechanisms [1]
- SAM: $4.4B -- 70% geographic coverage (NA/EU/APAC) × 70% segment fit for template-based workflows = 49% of TAM
- SOM Year 1: $2-4M -- 10K-20K paid users at $200 ARPU (assuming 5-10% freemium conversion)
- SOM Year 3: $15-30M -- 75K-150K paid users at $200 ARPU with enterprise uptake

Market sizing methodology relies on productivity software category analysis but lacks browser extension-specific data -- founder should validate these numbers with bottom-up user research.

5 Validation Experiments

#	Experiment	Cost	Time	Success Criteria
1	Landing page + r/Notion template post	$200	1 week	5%+ CTR from reddit traffic, 200+ email signups
2	Cold email to 100 remote workers	$0	3 days	15%+ response rate, 5+ problem validation calls
3	Fake door MVP (Chrome extension waitlist)	$100	2 weeks	10%+ install-intent rate from targeted ads
4	Manual task creation service (concierge MVP)	$0	1 week	5+ users willing to pay $20 for manual template creation
5	Competitor analysis via browser automation	$50	2 days	Identify 3+ workflow gaps in existing solutions

Distribution Channel Ranking

1. Community infiltration (r/Notion, r/productivity): These audiences actively seek workflow optimization and share tools. First test: Share 3 high-value templates weekly for 30 days, measure engagement and DM requests.

2. SEO for "[tool] + template" keywords: Users search "Notion task templates" and "Asana workflow templates" monthly. First test: Create landing pages for top 5 template keywords, measure organic traffic over 60 days.

3. Partnership with productivity YouTubers/creators: Creators need new tools to review and audiences trust their recommendations. First test: Reach out to 10 productivity channels with <100K subscribers, offer early access for review.

Cost-to-Validate vs Cost-to-Build

Validation: $350 + 3 weeks (5 experiments + basic landing page setup) Build: $75K-125K + 16-20 weeks (medium complexity browser extension with API integrations, based on provided budget/timeline estimates)

Validation cost is reasonable given the build investment required. The 20:1 ratio suggests thorough validation is essential before committing to development.

30 / 60 / 90-Day Validation Roadmap

Day 30 (Validation Gate):

- Complete all 5 validation experiments
- 500+ email signups from community engagement
- 10+ customer interviews with target users
- Identify 2-3 highest-value template categories
- Gate criteria: 40%+ of interviewed users express willingness to pay $9/month

Day 60 (MVP Scaffold):

- Chrome extension boilerplate with Notion API integration
- 3 core templates built and tested
- 50 beta users actively testing daily
- Basic analytics and feedback collection implemented
- Gate criteria: 20%+ daily active usage among beta users

Day 90 (First Revenue or Kill):

- Freemium model live with payment processing
- 10+ paying customers at $9/month minimum
- User feedback on feature prioritization for V2
- Chrome Web Store submission complete
- Decision point: Continue to Series A fundraising or pivot/kill based on unit economics

Kill Criteria

- If customer interviews reveal <30% willingness to pay any amount, the problem isn't painful enough for monetization
- If existing productivity tools add similar template functionality during our build phase, competitive advantage disappears

- If Chrome Web Store approval takes >60 days or requires major architecture changes, time-to-market advantage is lost
- If beta users show <15% weekly active usage after 30 days, product-market fit is insufficient
- If paid conversion rate falls below 2% after 90 days of freemium operation, unit economics don't support growth

Cold-Outreach Script (drop-in)

Email Subject: Quick question about your task management workflow

Hi [Name],

I noticed you're active in productivity communities and wondered if you experience this: when you're researching online and need to create a structured task in [Notion/Asana/Trello], do you find yourself losing context switching between browser tabs?

I'm building a Chrome extension that lets you create template-based tasks instantly from any webpage. Would you be interested in a 10-minute call to share what workflow friction you experience most?

Best, [Your Name]

LinkedIn DM: Hi [Name], saw your post about [SpecificPainPoint] in task management. Building something that might help - could I get 5 minutes of your thoughts on workflow automation? Thanks!

Landing-Page Copy (drop-in)

Hero H1: Stop Losing Ideas Between Browser Tabs

Sub-bullets:

- Create structured tasks in Notion, Asana, or Trello without leaving your current webpage
- Pre-built templates for research, client requests, and project ideas save 2-5 minutes per task
- Smart context capture automatically pulls page titles, URLs, and selected text into your tasks

CTA: Get Early Access + 50 Free Templates

What You Get:

- Chrome extension with one-click task creation
- 20+ pre-built templates for common workflows
- Direct integration with your existing productivity tools

Sources & References

1. [1] https://www.failory.com/interview/tactiq
2. [2] https://techcrunch.com/2022/09/14/magical-snags-35m-to-automate-browser-based-workflows/
3. [3] https://techcrunch.com/2022/05/10/workflow-documentation-service-tango-raises-14m/
4. [4] https://techcrunch.com/2022/01/20/ai-calendar-startup-motion-exits-stealth-with-13m-seed/

5. [5] https://todoist.com/pricing
6. [6] https://www.any.do/pricing
7. [7] https://ticktick.com/about/pricing
8. [8] https://market.us/report/ai-browser-market/
9. [9] https://trello.com/pricing
10. [10] https://asana.com/pricing

03 Product Requirements Document (PRD)

EXECUTIVE SUMMARY

QuickCreate tackles the universal productivity pain: losing context when creating tasks across multiple tools.

- Product: Browser extension for instant task creation using customizable templates
- Core Problem: Context-switching friction between research/communication and task capture
- Primary Users: Remote workers managing 3+ productivity tools simultaneously
- MVP Timeline: 4-6 weeks to Chrome extension with 3 core integrations
- Success Signal: 15% weekly active usage among pilot users within 60 days
- Revenue Model: Freemium with premium templates and advanced integrations at $9/month

Product Overview

Product Name: **QuickCreate Vision:** Make task creation as fast as bookmarking a webpage -- from anywhere on the web.

Platform: **Chrome browser extension (Manifest v3) Core Value Proposition:** Create structured tasks in your preferred productivity tool without leaving your current browser tab, using pre-built templates that capture context automatically.

Key Differentiators:

- Universal: works across Notion, Asana, Trello, ClickUp from any webpage
- Context-aware: auto-captures current page URL, selected text, timestamp
- Template-driven: reduces 2-5 minute task creation to 15 seconds

Build This FIRST -- the sharpest version of the MVP

The one-feature MVP: A Chrome extension popup that creates tasks in Notion using 3 pre-built templates ("Research Task," "Meeting Follow-up," "Bug Report") with auto-filled context from the current webpage.

Why this specific feature: Market validation shows 60/100 demand driven by workflow friction. Notion has the largest power-user base among knowledge workers who create 10+ tasks daily. Three templates cover 70% of context-switching scenarios (research -> task, meeting -> follow-up, bug discovery -> ticket). Auto-context capture (URL, selected text) eliminates the copy-paste workflow that creates 2-5 minute delays.

The cut list -- what NOT to build in v1:

- Custom template builder -- adds 3 weeks of dev, users will adapt to pre-built templates initially
- Multi-tool sync -- complex data mapping, ship with Notion-only first
- Team sharing features -- B2B complexity, focus on individual workflow first
- Advanced formatting (rich text, attachments) -- plain text captures 90% of use cases
- Mobile app -- browser extension validates concept faster, mobile comes later
- AI-powered template suggestions -- interesting but not core to workflow improvement
- OAuth login -- email/password is sufficient, OAuth adds 2 weeks for marginal UX gain

Build order for v1:

1. Week 1: Static popup with hardcoded Notion API connection and one "Research Task" template
2. Week 2: Add context capture (current URL, page title, selected text)
3. Week 3: Add two more templates ("Meeting Follow-up," "Bug Report") with different field mappings
4. Week 4: Basic settings page for Notion workspace/database selection
5. Week 5: Error handling and success feedback in popup
6. Week 6: Chrome Web Store submission and packaging

Decision rule for what to add next: If 40% of pilot users (minimum 50 people) use the extension 2+ times per week for 3 consecutive weeks, add Asana integration. If usage drops below 20% weekly active in month 2, pivot to different context capture approach or kill project.

User Personas & Stories

Primary Persona: Sarah, Remote Product Manager

- Age: 32, works for 50-person SaaS company
- Tools: Notion (project docs), Asana (task tracking), Slack, Gmail, Chrome (research)
- Pain: Constantly switching between browser research and Asana task creation, losing context and momentum
- Workflow: Researches competitor features -> needs to create tasks for engineering team with specific requirements and links

Secondary Persona: Mike, Freelance Marketing Consultant

- Age: 28, manages 4-6 clients simultaneously
- Tools: Trello (client projects), Notion (knowledge base), Google Docs, client communication tools
- Pain: Context-switching between client research, communication, and task creation causes 1-2 hours of lost productivity daily
- Workflow: Client email -> research solution -> create action items with deadlines and context

Tertiary Persona: Lisa, Engineering Team Lead

- Age: 35, leads 8-person engineering team
- Tools: Linear (bug tracking), Notion (documentation), GitHub, Slack

• Pain: Discovering bugs or improvement opportunities while browsing but friction in creating proper tickets with context

• Workflow: Code review -> identifies issue -> needs detailed ticket with repo links and specifications

User Stories:

1. As Sarah, I want to create an Asana task from a competitor's feature page so that I can track research with proper context and links

2. As Sarah, I want to use a "Research Task" template so that I capture consistent information (URL, description, priority) every time

3. As Mike, I want to create a Trello card from a client email so that I don't lose action items in my inbox

4. As Mike, I want templates for different client work types so that I include the right fields and due dates automatically

5. As Lisa, I want to create a Linear issue while reviewing documentation so that bugs are logged with proper context immediately

6. As all personas, I want to see confirmation that my task was created so that I know the system worked

7. As all personas, I want to select which workspace/project receives the task so that items go to the right place

8. As power users, I want keyboard shortcuts to trigger templates so that task creation becomes muscle memory

9. As team leads, I want to assign created tasks to team members so that delegation happens immediately

10. As all personas, I want to capture selected text automatically so that context is preserved without copy-paste

11. As all personas, I want the extension to remember my preferences so that subsequent task creation is faster

12. As Sarah, I want to create multiple related tasks from one research session so that complex projects are broken down immediately

13. As Mike, I want to set default project/lists for different domains so that client work auto-routes correctly

14. As all personas, I want to edit task details before creation so that I can refine information quickly

15. As power users, I want to see recent tasks created so that I can verify and potentially duplicate similar work

Acceptance Criteria for Top 5 Stories:

Story 1 (Sarah - Asana from competitor page):

• Extension popup opens from any webpage

• "Research Task" template pre-populates with current URL and page title

• User can edit task title, description, and select Asana project

• Task appears in correct Asana project within 30 seconds

- Success message appears in extension popup

Story 2 (Sarah - Research Task template):

- Template includes fields: Task Title, Description, URL (auto-filled), Priority dropdown, Project selector
- Auto-fills current page URL and title as starting point
- Saves to user's default Asana workspace and selected project
- Template consistently formats tasks for team recognition

Story 6 (Confirmation feedback):

- Green checkmark and "Task created successfully" message appear in popup
- Link to created task opens in new tab when clicked
- Error states show clear messaging for API failures or missing permissions
- Popup remains open for 3 seconds then auto-closes on success

Story 10 (Auto-capture selected text):

- If user has text selected on page when opening extension, it auto-populates in description field
- Selected text is clearly marked/quoted in the description
- User can edit or remove the captured text before task creation
- Works with up to 500 characters of selected text

Story 11 (Remember preferences):

- Extension remembers last-used project/workspace selection
- Default template selection persists between sessions
- User preferences sync across Chrome instances when logged into Chrome
- Settings are preserved even after extension updates

Feature Specification

MVP Features (Must-Have for Launch)

Extension Popup Interface

- Description: Clean, minimal popup (350px x 500px) with template selector and task creation form
- User Story: As Sarah, I want to quickly access task creation without navigating away from my research
- Priority: P0 (launch blocker)
- Complexity: Medium

Notion API Integration

- Description: Direct API connection to create tasks in user's Notion workspace with proper authentication
- User Story: As Sarah, I want tasks to appear immediately in my Notion workspace

- Priority: P0 (launch blocker)
- Complexity: High

Context Auto-Capture

- Description: Automatically captures current page URL, title, and selected text when extension is activated
- User Story: As all personas, I want context preserved without manual copy-paste
- Priority: P0 (launch blocker)
- Complexity: Low

Three Core Templates

- Description: Pre-built templates for "Research Task," "Meeting Follow-up," and "Bug Report" with appropriate field mappings
- User Story: As Mike, I want templates for different work types with consistent structure
- Priority: P0 (launch blocker)
- Complexity: Medium

Workspace/Database Selection

- Description: Dropdown to select target Notion workspace and database for task creation
- User Story: As all personas, I want control over where tasks are created
- Priority: P1 (launch with)
- Complexity: Medium

Phase 2 Features (Post-Launch)

Asana Integration (Month 2)

- Second productivity tool integration based on user feedback
- Similar template structure adapted for Asana's task format

Custom Template Builder (Month 2)

- Basic interface to create custom templates with field mapping
- Limited to 5 custom templates per user initially

Keyboard Shortcuts (Month 3)

- Configurable hotkeys to trigger extension popup and specific templates
- Default: Ctrl+Shift+T for popup, Ctrl+Shift+1/2/3 for templates

Recent Tasks View (Month 3)

- Mini-dashboard showing last 10 created tasks with links
- Quick duplicate/edit functionality for similar tasks

Future Roadmap Features

Team Template Sharing (Month 4-6)

- Share custom templates across team members

- Team-specific template libraries

AI-Powered Context Enhancement (Month 4-6)

- Auto-suggest task titles based on page content
- Smart categorization and priority suggestions

Multi-Tool Sync (Month 6+)

- Create related tasks across multiple tools simultaneously
- Cross-platform task relationship mapping

Mobile Companion App (Month 6+)

- iOS/Android app for task creation from mobile browsers
- Sync with desktop extension preferences

Information Architecture

```
QuickCreate Extension
+-- Popup Interface (Main)
|   +-- Template Selector (3 buttons)
|   +-- Task Creation Form
|   |   +-- Title Field
|   |   +-- Description Field (with auto-context)
|   |   +-- Project/Database Selector
|   |   +-- Create Task Button
|   +-- Success/Error Feedback
+-- Settings Page
|   +-- Account Connection (Notion OAuth)
|   +-- Default Workspace Selection
|   +-- Template Preferences
|   +-- Keyboard Shortcuts Config
+-- Options Page (Chrome Extension Settings)
|   +-- Privacy Settings
|   +-- Extension Permissions
|   +-- Help Documentation
+-- Background Service Worker
    +-- API Communication
    +-- Context Capture Logic
    +-- User Preferences Storage
```

Key Screen Descriptions:

Popup Interface (350px × 500px):

- Header: QuickCreate logo + current page favicon/title
- Template buttons: 3 prominent buttons for core templates
- Dynamic form: Changes based on selected template
- Footer: Settings icon, help link, success/error status

Settings Page:

- Clean single-page interface accessible via popup settings icon
- OAuth connection status with connect/disconnect buttons

- Workspace dropdown with refresh capability
- Template preference toggles and defaults

Functional Requirements

Task Creation Flow:

- Input: User clicks extension icon -> selects template -> fills form -> clicks create
- Processing: Extension captures page context -> validates form data -> calls Notion API -> stores user preferences
- Output: Task created in Notion + success confirmation + link to created task

Context Capture Logic:

- Input: Extension activation triggers context scan
- Processing: Captures document.title, window.location.href, window.getSelection().toString()
- Output: Auto-populated form fields with captured data
- Business Rule: Selected text limited to 500 characters, URL validation for proper formatting

Template Processing:

- Input: User selects template type
- Processing: Load template configuration -> map fields to Notion properties -> apply context data
- Output: Populated form matching template structure
- Business Rule: Each template maps to specific Notion database properties

API Integration Flow:

- Input: Completed task form with user authentication
- Processing: OAuth token validation -> API payload construction -> POST request to Notion
- Output: Created task object with database ID and URL
- Error Handling: Rate limiting (429), auth errors (401), API failures (500) with user-friendly messages

User Preferences:

- Input: User selections and configurations
- Processing: Chrome storage API with sync across devices
- Output: Persisted settings for future sessions
- Business Rule: Preferences stored locally, no server-side user data

Non-Functional Requirements

Performance Targets:

- Extension popup opens in <200ms after icon click
- Task creation completes within 3 seconds of form submission

- Context capture processes in <100ms on popup open
- Memory usage stays under 50MB for background service worker

Scalability Requirements:

- Support 10,000 concurrent active users in Month 1
- Handle 50,000 task creations per day by Month 6
- Chrome Web Store supports 100,000+ user installs

Security Requirements:

- OAuth 2.0 for Notion API authentication
- No plaintext storage of user credentials
- HTTPS-only API communication
- Manifest v3 compliance for Chrome security model
- User data encrypted in Chrome storage

Accessibility Requirements:

- WCAG 2.1 AA compliance for popup interface
- Keyboard navigation support for all interactive elements
- High contrast mode compatibility
- Screen reader compatibility for form fields and buttons

Offline Capability:

- Extension works offline for context capture and form completion
- Queue task creation requests when offline, sync when connection returns
- Local storage for user preferences and template configurations

Data Requirements

Core Data Entities:

Entity	Properties	Storage Location
User Settings	workspace_id, default_database, template_preferences	Chrome Storage (sync)
Templates	name, fields[], notion_mapping{}, is_custom	Chrome Storage (local)
Task Context	url, title, selected_text, timestamp, template_used	Temporary (not stored)
API Credentials	oauth_token, refresh_token, workspace_info	Chrome Storage (local, encrypted)

User Data Collection:

- Notion workspace authorization (OAuth scope)
- Template usage preferences

- Default project/database selections
- Extension usage analytics (anonymous)

Third-Party Integrations:

- Notion API: Database reads, page creation, workspace metadata
- Chrome Storage API: User preferences and credentials
- Chrome Tabs API: Current page context and URL access

Data Retention & Privacy:

- No server-side storage of user tasks or content
- OAuth tokens stored locally in encrypted Chrome storage
- Usage analytics aggregated and anonymized
- User can revoke access and clear all stored data via settings

Integration Requirements

Notion API Integration:

- Authentication: OAuth 2.0 with scopes: `read`, `write`
- Endpoints:
  - `GET /v1/databases` (workspace database list)
  - `POST /v1/pages` (task creation)
  - `GET /v1/users/me` (user workspace info)
- Rate Limits: 3 requests per second, respect 429 responses
- Webhooks: Not required for MVP

Chrome Extension APIs:

- Tabs API: Access current tab URL and title
- Storage API: Sync user preferences, store credentials locally
- Action API: Extension popup trigger and badge updates
- Scripting API: Inject content scripts for text selection capture

Analytics Integration (Optional for MVP):

- Mixpanel or Amplitude: Extension usage events
- Events: popup_opened, template_selected, task_created, error_occurred
- Privacy: No PII collection, anonymous user IDs only

Future Integrations (Post-MVP):

- Asana API: Similar OAuth flow and task creation endpoints
- Trello API: Card creation with board/list selection
- Linear API: Issue creation for bug reports template

Success Metrics

Primary KPIs:

- Weekly Active Users (WAU): Target 40% of installed users active weekly by Month 2
- Task Creation Rate: Average 5+ tasks created per active user per week
- Retention Rate: 60% of users who create 3+ tasks return the following week
- Template Usage Distribution: Each core template used by 30%+ of active users monthly

Secondary KPIs:

- Extension Rating: Maintain 4.2+ stars on Chrome Web Store
- API Success Rate: 95%+ successful task creation (excluding user errors)
- Support Ticket Volume: <2% of users contact support monthly
- Conversion to Premium: 8-12% freemium to paid conversion within 90 days

North Star Metric: Total tasks created per week -- combines user growth, engagement, and product-market fit into single measure

Measurement Methods:

- Chrome Extension Analytics: Built-in user metrics and crash reporting
- Mixpanel Events: Custom event tracking for user actions and funnel analysis
- Notion API Logs: Success/failure rates for task creation requests
- User Surveys: Monthly NPS and feature feedback via in-app prompts
- Chrome Web Store Analytics: Install rates, ratings, and review sentiment tracking

PRO TIP
Start measuring from day 1 of alpha testing -- even 10 power users give better signal than waiting for 1,000 casual users.

Decision Checklist

1. Validate Notion integration first -- Build basic API connection and create one test task in your own workspace before any UI work
2. Interview 5 power users this week -- Find people who create 10+ tasks daily across multiple tools; validate the context-switching pain
3. Map competitor extensions -- Install Momentum Dash, Todoist's extension, and 2-3 productivity extensions to understand UX patterns
4. Prototype popup UI in Figma -- Design the 350px × 500px popup with all three templates before coding begins
5. Set up Chrome extension developer account -- $5 registration fee, takes 2-3 days for approval, required for testing
6. Define the kill criteria -- Write down specific usage thresholds that would cause you to pivot or shut down
7. Plan beta user recruitment -- Identify 25-50 potential beta testers from your network before launch week

Sources & References

1. [1] Productivity software market research and validation data from Unbuilt Lab platform
2. [2] Chrome Web Store analytics and browser extension market data
3. [3] Notion API documentation and integration requirements
4. [4] Manifest v3 specifications and Chrome extension development guidelines

04 Technical Architecture Blueprint

Architecture Overview

This is a client-heavy architecture with minimal backend, designed for speed and cost-efficiency.

Pattern: Client-heavy serverless with optional lightweight backend

The architecture centers around a Chrome extension with local data processing, connecting directly to third-party APIs (Notion, Asana) through OAuth flows. No custom backend for MVP -- authentication and data flow happen entirely client-side with Chrome's storage APIs handling state management.

Component Flow:

1. Extension Frontend (popup + background service worker) handles all user interaction and template logic
2. Chrome Storage APIs manage user preferences, templates, and OAuth tokens locally
3. Direct API Integration to productivity tools (Notion first, Asana second) via their REST APIs
4. Optional Analytics Service (serverless function) for usage tracking without PII

Key Architectural Decisions:

• No custom backend initially -- reduces infrastructure costs from $500/mo to $50/mo and eliminates scaling complexity

• Chrome Storage over external DB -- simpler development, better offline capability, no data privacy concerns

• Direct API integration -- faster task creation (no middleware latency), fewer failure points

• Serverless analytics only -- can scale from 0 to 100K users without infrastructure changes

Technology Stack Recommendation

Layer	Technology	Rationale
Frontend	TypeScript + React 18 + Vite	Type safety critical for API integrations; React for complex popup state; Vite for fast dev builds
Extension Framework	Plasmo Framework	Built for Manifest v3; handles TypeScript, React, and build pipeline; Chrome/Firefox compatibility
Backend	None for MVP	Direct API calls eliminate backend complexity; add serverless functions only when needed
Database	Chrome Storage API	Free, offline-capable, encrypted; perfect for user preferences and templates
Auth	OAuth 2.0 (direct)	Required by Notion/Asana APIs; no custom auth server needed

Hosting	Chrome Web Store	Zero hosting costs; built-in update mechanism
CI/CD	GitHub Actions	Free for open source; integrates with Chrome Web Store publishing

Version Recommendations:

- Node.js 18+ (for build tools)
- TypeScript 5.0+
- React 18.2+ (concurrent features for better UX)
- Plasmo 0.76+ (stable Manifest v3 support)

Database Design

Local storage design using Chrome Storage API -- no traditional database needed.

Storage Entities:

Storage Key	Structure	Type	Sync/Local
`user_settings`	`{ default_workspace, default_database, preferred_templates[] }`	Object	Sync
`oauth_tokens`	`{ notion: {access, refresh}, asana: {access, refresh} }`	Object	Local
`templates`	`{ id, name, fields[], platform_mapping, is_custom, created_at }`	Array	Local
`usage_stats`	`{ tasks_created, last_active, template_usage{} }`	Object	Local

Data Relationships:

- Templates reference platform-specific field mappings (Notion property IDs, Asana custom fields)
- User settings link to workspace/database IDs from connected services
- No foreign keys or complex relationships -- simple object storage

Storage Limits:

- Chrome Sync Storage: 100KB total, 8KB per item (perfect for user settings)
- Chrome Local Storage: Unlimited (templates and tokens stored here)

Migration Strategy:

- Chrome extension auto-update handles data migrations
- Version-based migration scripts in background service worker

- Graceful fallbacks for missing or corrupted data

API Design

External API integration strategy -- no custom APIs for MVP.

Integration Pattern: Direct client-to-service API calls with OAuth 2.0

Key External APIs:

Service	Endpoint	Purpose	Rate Limit
Notion	`POST /v1/pages`	Create tasks	3/sec
Notion	`GET /v1/databases/{id}`	Fetch database schema	3/sec
Notion	`GET /v1/databases`	List user databases	3/sec
Asana	`POST /api/1.0/tasks`	Create tasks (Phase 2)	1500/hour

Authentication Flow:

1. User clicks "Connect Notion" in extension popup
2. Extension opens OAuth redirect in new tab: `https://api.notion.com/v1/oauth/authorize`
3. User grants permissions, Notion redirects with auth code
4. Extension exchanges code for access token via content script
5. Tokens stored in Chrome Local Storage (encrypted)

Error Handling:

- Retry logic with exponential backoff for rate limits
- Graceful degradation when APIs are unavailable
- Clear user messaging for auth failures or permission issues

No Custom APIs Needed:

- Template logic handled client-side in TypeScript
- User preferences sync via Chrome Storage
- Analytics (optional) via simple POST to serverless endpoint

Security Architecture

Security-first design leveraging Chrome's built-in protections.

Authentication: OAuth 2.0 with PKCE (Proof Key for Code Exchange)

- No client secrets stored -- use PKCE flow for public clients
- Access tokens stored in Chrome Storage Local (automatically encrypted)
- Refresh tokens for long-term access, rotated on each use

Authorization Model: Service-level permissions

- Extension requests minimum required scopes from each service
- Users can revoke access through service provider dashboards
- No custom permission system -- rely on Notion/Asana's built-in controls

Data Encryption:

- In Transit: All API calls over HTTPS (enforced by Manifest v3)
- At Rest: Chrome Storage automatically encrypts local data
- Tokens: Additional encryption layer using Web Crypto API before storage

Input Validation:

- TypeScript interfaces for all API payloads
- Zod schemas for runtime validation of user inputs
- Content Security Policy prevents script injection
- Sanitize all user text before sending to external APIs

OWASP Top 10 Mitigations:

- Injection: Parameterized API calls, no dynamic query building
- Broken Auth: OAuth 2.0 + PKCE, no custom auth logic
- Data Exposure: No sensitive data logging, minimal data collection
- XML External Entities: N/A (no XML processing)
- Access Control: Chrome extension permissions model
- Security Misconfiguration: Content Security Policy, HTTPS-only

Infrastructure & Deployment

Minimal infrastructure leveraging Chrome Web Store and serverless functions.

Hosting Strategy:

- Primary: Chrome Web Store (free distribution, auto-updates)
- Analytics: Vercel serverless functions (free tier: 100GB bandwidth)
- Website/Landing: Vercel static hosting (free)

Environment Setup:

- Development: Local Chrome extension in developer mode
- Staging: Chrome Web Store private listing for team testing
- Production: Chrome Web Store public listing

CI/CD Pipeline (GitHub Actions):

```
1. Pull request -> Run tests, lint, type check
2. Merge to main -> Build extension, upload to Chrome Web Store
3. Manual approval -> Publish to production
```

Monitoring:

- Extension: Chrome extension error reporting (built-in)

- APIs: Rate limit monitoring via custom analytics
- User feedback: Chrome Web Store reviews + optional in-app feedback

Monthly Infrastructure Costs:

Service	Tier	Cost	Purpose
Chrome Web Store	Developer fee	$5 (one-time)	Extension hosting
Vercel	Hobby	$0	Analytics functions + landing page
Domain	.com registration	$12/year	Marketing site
Total Month 1	**$1/mo**	After initial $5 registration	
Total Month 12	**$1/mo**	No scaling costs until 100K+ users	

Scalability Plan

Designed to scale from 0 to 100K users without infrastructure changes.

Current Architecture Handles: 50,000 concurrent users (Chrome Storage limitations)

Scaling Triggers:

- 10K users: Add basic analytics to understand usage patterns
- 50K users: Consider backend for advanced features (team sharing, sync)
- 100K+ users: Migrate to hybrid architecture with optional cloud sync

Database Scaling:

- Chrome Storage scales with user count (no shared database)
- Each user gets isolated 10MB+ storage space
- No database scaling issues until backend introduction

API Rate Limiting Strategy:

- Client-side: Implement request queuing with 3/sec limit for Notion
- Exponential backoff: 1s, 2s, 4s delays on 429 responses
- User education: Clear messaging about rate limits in UI

Caching Strategy:

- Template definitions: Cache in Chrome Storage (rarely change)
- Database/workspace info: Cache for 1 hour, refresh on error
- User context: No caching (always current tab data)

CDN: Not needed -- Chrome Web Store handles global distribution

Third-Party Services

Service	Provider	Purpose	Cost Tier	Alternative
OAuth Provider	Notion/Asana	User authentication	Free	Auth0 ($25/mo, overkill)
Analytics	Mixpanel	Usage tracking	Free (1000 MTU)	PostHog (self-hosted)
Error Tracking	Sentry	Bug monitoring	Free (5K errors/mo)	Chrome extension console
Email	ConvertKit	User onboarding	Free (1K subscribers)	Mailgun ($35/mo)
Domain	Namecheap	Marketing site	$12/year	None (use GitHub pages)

> **PRO TIP**
> Don't add analytics until you have 1K+ users -- early-stage usage data is noise, and you'll spend more time configuring dashboards than building features.

Development Environment Setup

Required Tools:

```
Node.js 18+ (LTS)
Chrome Browser (latest)
VS Code + Chrome Extension tools
Git + GitHub CLI
```

Local Development Setup:

```
1. git clone <repo>
2. npm install
3. npm run dev
4. Chrome -> Extensions -> Developer mode -> Load unpacked
5. Point to dist/ folder
```

Environment Variables:

```
# .env.development
NOTION_CLIENT_ID=<from Notion developer portal>
NOTION_REDIRECT_URI=chrome-extension://<extension-id>/auth
MIXPANEL_TOKEN=<analytics token>
```

Code Quality Stack:

- ESLint + Prettier: Code formatting and basic linting
- TypeScript strict mode: Catch integration errors early
- Vitest: Unit tests for template logic and API integrations
- Playwright: E2E tests for extension workflows
- Husky: Pre-commit hooks for formatting and tests

Implementation Traps

Trap 1: Don't build a custom OAuth server

- The trap: "I'll handle OAuth myself to avoid vendor lock-in and save on third-party costs"
- When it bites: Week 3, when you realize OAuth 2.0 + PKCE implementation takes 2 weeks and you still need to handle refresh tokens, revocation, and security audits
- The cheaper avoid: Use direct OAuth to Notion/Asana APIs -- it's free, more secure, and required anyway. Skip the middleware complexity.

Trap 2: Don't over-engineer template storage with a backend database

- The trap: "Users need cloud sync for templates, so I'll build a backend with PostgreSQL for template sharing"
- When it bites: Month 2, when you're debugging user auth, database migrations, and sync conflicts instead of shipping template features users actually want
- The cheaper avoid: Chrome Storage handles 99% of use cases perfectly. Add backend sync only after 10K+ users explicitly ask for team template sharing.

Trap 3: Don't try to support all productivity tools at launch

- The trap: "I'll integrate with Notion, Asana, Trello, ClickUp, and Monday.com for maximum market appeal"
- When it bites: Week 4, when each integration has different auth flows, field mapping, and rate limits -- you're maintaining 5 integrations poorly instead of 1 excellently
- The cheaper avoid: Notion only for MVP. Add Asana in month 2 only if users specifically request it. One integration done well beats five done poorly.

Trap 4: Don't build complex template logic with visual editors

- The trap: "Users need a drag-and-drop template builder like Notion or Zapier"
- When it bites: Month 3, when you've spent 6 weeks building a form builder that works for 3 use cases but breaks on the 4th
- The cheaper avoid: Ship 3 hardcoded templates (Research, Meeting, Bug Report) that handle 80% of use cases. Add custom templates as simple JSON editing in month 4+.

Trap 5: Don't optimize for cross-browser compatibility from day 1

- The trap: "I'll use a cross-browser extension framework to support Chrome, Firefox, and Safari simultaneously"
- When it bites: Week 2, when Firefox's extension API differences force you to maintain separate code paths, and Safari's restrictions break your OAuth flow entirely
- The cheaper avoid: Chrome-only for MVP using Plasmo. Firefox support in month 4 if Chrome version proves viable. Safari isn't worth the complexity until 50K+ users.

Trap 6: Don't add advanced context capture (screenshot, full page content) early

- The trap: "I'll capture screenshots and full page HTML to give users rich context with their tasks"
- When it bites: Week 5, when you discover Chrome's permission model requires broad "access all websites" permissions, killing your security story and install rates
- The cheaper avoid: URL + title + selected text only. This covers 90% of context needs without scary permissions. Rich context is a month 6+ feature after users trust the extension.

05 Go-To-Market & Growth Strategy

EXECUTIVE SUMMARY

QuickCreate's GTM centers on capturing productivity power users through tactical community infiltration and feature-differentiated positioning.

- Launch Type: Product-Led Growth with community-seeded distribution
- Core Bet: Target the 15-20% of productivity tool users who actively customize their workflows -- they'll evangelizemicroptasks efficiency to their teams
- Timeline: 4-week pre-launch build-up -> 2-week launch blitz -> 6-month growth acceleration
- Key Risk: Chrome Web Store discovery is brutal for new extensions -- need owned distribution from day one

GTM Strategy Overview

Launch Strategy: Product-Led Growth with tactical community seeding. Free tier drives adoption, power features convert to paid, satisfied users become word-of-mouth amplifiers within their productivity tool ecosystems.

Positioning Statement: "The universal task capture extension that works with your existing tools -- stop losing ideas between browser tabs."

Key Messaging Framework:

- Headline: "Capture tasks instantly from any website -- no more copy-paste between tools"
- Subhead: "One-click templates that sync to Notion, Asana, Trello, and 15+ productivity platforms"
- Supporting Points:
    1. "Save 2-5 minutes per task with pre-built templates for research, client requests, and project ideas"
    2. "Works wherever you browse -- turn any webpage, email, or document into structured tasks"
    3. "Smart context capture -- automatically pulls page title, URL, and selected text into your tasks"

The Founder's Unfair Distribution Edge

Channel Asymmetry: Productivity Tool Community Infiltration

The asymmetry is hyper-focused infiltration of productivity tool communities where power users hang out complaining about workflow friction. Target communities:

- r/Notion (1.1M members) -- users constantly sharing template workflows and automation hacks
- r/productivity (1.8M members) -- workflow optimization obsessives who test new tools monthly
- Notion Facebook Groups (~200K combined) -- template sharers and workflow optimizers
- Indie Hackers productivity channels -- startup founders managing multiple tools simultaneously

Conversion Strategy: Founder becomes "the template guy" by sharing 15-20 high-value task templates (client onboarding, content planning, bug tracking) as free resources with light QuickCreate mentions. Power users who save time will naturally ask "how do I automate this?" -- that's the QuickCreate introduction moment.

The key: most productivity extensions spam these communities with feature lists. Instead, lead with free value (templates) that solve immediate problems, building trust before the pitch.

Target: 100 paying customers in 6 months through 3,000 community template downloads converting at 3-4%.

Pre-Launch Phase (Weeks 1-4)

Landing Page Strategy:

- Hero: Interactive demo showing task creation from a live webpage in 8 seconds
- Social Proof: "Join 2,847 productivity enthusiasts testing QuickCreate" (update weekly)
- Copy Direction: Problem-agnostic ("Stop losing ideas") vs feature-focused ("Browser extension with templates")
- CTA: "Get Early Access + 50 Free Templates"

Waitlist Campaign:

- Week 1-2: Seed with founder's existing network (LinkedIn, Twitter followers)
- Week 3-4: Share in 5-6 productivity communities with template lead magnets
- Target: 800-1,200 waitlist signups

Beta User Recruitment:

- Where: r/Notion template creators, Indie Hackers community, productivity YouTube comments
- How: "Looking for 50 power users to test universal task templates -- get lifetime Pro free"
- Target: 50-75 beta users across 5-6 productivity tools

Content Creation:

- Blog Posts:
  - "15 Task Templates That Save Me 10 Hours/Week"
  - "The Hidden Cost of Context-Switching Between Productivity Tools"
- Social Media: Daily template shares on Twitter with light QuickCreate branding
- YouTube: Partner with 2-3 productivity channels for template tutorial videos

Launch Phase (Weeks 5-6)

Launch Timeline:

- Tuesday 6 AM PST Product Hunt -- optimal for tech audience, avoid Monday competition
- Wednesday Hacker News Show HN -- "QuickCreate: Universal task templates for any website"
- Thursday-Friday Reddit -- r/productivity, r/Notion, r/SideProject with different angles
- Social Media Sequence: 7-day announcement campaign across Twitter, LinkedIn

Assets Needed:

- Product Hunt: 5 preview GIFs, maker comment strategy, hunter outreach
- HN: Technical demo video, architecture explanation for developer credibility
- Press Kit: Logo variations, founder photos, one-page fact sheet

Media Outreach:

- Tier 1: TechCrunch (productivity tools beat), Product Hunt blog
- Tier 2: The Verge, Lifehacker, Fast Company (productivity sections)
- Tier 3: Productivity newsletters (Morning Brew, Lenny's Newsletter productivity issues)

Post-Launch Growth (Months 2-6)

Organic Channels

SEO Strategy:

- Primary Keywords: "task template extension" (2,400/mo), "productivity browser extension" (1,900/mo)
- Long-tail: "notion task templates chrome extension" (400/mo), "asana quick add extension" (200/mo)
- Content Calendar:
  - Month 2: Template comparison guides ("Notion vs Asana Templates")
  - Month 3-4: Workflow optimization tutorials
  - Month 5-6: Integration deep-dives and power user case studies

Social Strategy:

- Twitter: Daily template shares, reply to productivity pain point tweets
- LinkedIn: Weekly long-form posts on workflow optimization
- YouTube: Monthly "Template Spotlight" series with 3-5 productivity YouTubers

Paid Channels

Channel	Monthly Budget	Target CPA	Expected ROAS	Rationale
Google Ads	$3,000	$25-35	3.5-4.0x	High-intent keywords like "task template software"

Twitter Ads	$1,500	$20-30	4.0-5.0x	Promoted tweets in productivity hashtags
Productivity Newsletter Sponsorships	$2,000	$15-25	5.0-6.0x	High-quality audience pre-qualified for tools

A/B Testing Plan:

- Ad Creative: Template demo videos vs static screenshots
- Landing Pages: Free template download vs direct Chrome Web Store
- Targeting: Broad productivity vs specific tool users (Notion, Asana)

Viral & Referral

Referral Program: "Share QuickCreate, Get Premium Templates Free"

- Give 1 month Pro access for every 3 referral installs
- Referred users get starter template pack worth $15

Viral Mechanics:

- Template sharing within extension -- "Send this template to a teammate"
- Branded template exports -- subtle QuickCreate footer on shared templates

Partnership Opportunities:

- Notion Template Creators: Revenue share on premium template packs
- Productivity YouTubers: Affiliate commission for featuring QuickCreate templates
- SaaS Tool Integrations: Co-marketing with smaller productivity tools seeking Chrome extension presence

Pricing Strategy

Recommended Model: Freemium with template-tier upgrades

Tier	Price	Templates Included	Key Features
Free	$0/mo	5 basic templates	Chrome only, 2 tool integrations
Pro	$8/mo	50+ templates + custom	All browsers, unlimited tools, team sharing
Team	$20/user/mo	Everything + team templates	Admin dashboard, usage analytics, SSO

Price Point Justification:

- $8/mo sits below Notion ($8) and Todoist ($4) but above simple extensions ($3)
- Template value justifies premium -- comparable to design template subscriptions ($10-15/mo)
- Team tier captures growing remote work collaboration needs

Pricing Page Layout:

- Feature comparison table with competitor pricing context
- Template preview gallery showing Pro tier value
- "Most Popular" badge on Pro tier (standard conversion optimization)

Conversion Optimization

Onboarding Flow:

1. Install Confirmation: "QuickCreate is ready -- let's create your first template"
2. Tool Connection: "Connect your Notion/Asana account (takes 30 seconds)"
3. Template Selection: "Choose 3 starter templates for your workflow"
4. First Task Creation: "Try it now -- create a task from this welcome page"
5. Success State: "Perfect! Your task was added to [connected tool]"

Activation Metrics:

- Primary: User creates 3+ tasks within 7 days (correlates to retention)
- Secondary: User connects 2+ productivity tools within 14 days

Retention Strategy:

- Week 1: Welcome email series with template tutorials
- Week 2-4: Usage-triggered tips ("You've saved 15 minutes with templates this week!")
- Monthly: New template releases and power user spotlights

Churn Reduction:

- At-risk signals: No usage in 7 days triggers re-engagement email
- Win-back campaign: "Missing your templates? Here's what's new..."
- Exit survey: Capture cancellation reasons for product improvement

Key Metrics & Targets

Metric	Month 1	Month 3	Month 6
Chrome Installs	2,500	8,500	25,000
Active Users (7-day)	800	3,400	12,500
Free-to-Pro Conversion	2.5%	4.5%	6.0%
MRR	$500	$3,800	$15,000
Monthly Churn Rate	15%	8%	5%

Leading Indicators:

- Template usage frequency (target: 8+ uses/month per active user)
- Tool integration connections (target: 2.3 avg integrations/user)
- Template sharing rate (target: 15% of Pro users share templates monthly)

Marketing Budget Estimate

Channel	Monthly Budget	Expected CAC	Expected ROI
Google Ads	$3,000	$28	4.2x
Twitter Ads	$1,500	$24	4.8x
Newsletter Sponsorships	$2,000	$18	5.5x
Content Creation	$1,200	N/A	Organic multiplier
Community Management	$800	$12	6.0x+
Influencer Partnerships	$1,000	$22	4.0x

Total Monthly Marketing Budget: $9,500

Budget Allocation Logic:

- 60% paid acquisition (proven channels first)
- 25% content and community (long-term organic growth)
- 15% partnerships and experiments (future channel discovery)

PRO TIP

Start with 50% of this budget ($4,750/month) for months 2-3, then scale up based on unit economics validation. Most browser extensions fail by overspending on acquisition before proving product-market fit.

KEY INSIGHT

The real competitive moat isn't the extension technology -- it's the template ecosystem and community. Invest early in template creation and power user cultivation over pure download metrics.

Decision Checklist

1. Validate your template edge -- Build 20 high-quality templates before launch. Test them with 10 beta users who currently use manual workflows. Get specific feedback on time saved.
2. Secure your community foothold -- Join 5-6 productivity communities and contribute valuable templates/advice for 2-3 weeks before any QuickCreate mentions. Build trust first.
3. Set up measurement infrastructure -- Install Mixpanel or Amplitude for usage tracking, Hotjar for onboarding flow analysis, and Google Analytics for web conversion funnels.
4. Test Chrome Web Store optimization -- Research keyword rankings for "task template" and "productivity extension" -- optimize your store listing for discovery since organic traffic is limited.
5. Plan your launch week tactical sequence -- Schedule Product Hunt submission, prep HN technical demo, and coordinate with 3-4 productivity influencers for launch day coverage.
6. Build your referral system early -- Code the template sharing functionality into your MVP. Word-of-mouth is the highest-ROI channel for productivity tools.

7. Establish partnership pipeline -- Reach out to 5-6 Notion template creators and 2-3 productivity newsletters before launch. Partnerships take 4-6 weeks to materialize, so start early.

06 Project Execution & Delivery Roadmap

EXECUTIVE SUMMARY

QuickCreate is a medium-complexity browser extension with tight focus and clear MVP scope -- ship in 8-10 weeks with a lean team.

- Build now. Demand signal (60/100) is real; the gap (40/100) is solvable; feasibility is high (70/100). This is a de-risked bet in the medium-complexity range.
- Team shape: 1 senior engineer + 1 product/design person (can be founder-wearing-hats at start). Browser extensions don't need front-end/back-end separation; no full-time DevOps needed.
- Total timeline: 8-10 weeks from kickoff to Chrome Web Store launch. Week 1-2 for setup and auth, Week 3-6 for MVP features, Week 7-8 for polish/testing, Week 9-10 for launch prep.
- Budget envelope: $16K-$24K (within your medium range). Breaks down: $12K-$16K labor, $2K-$3K third-party services, $2K-$5K buffer. No infrastructure costs (Chrome Web Store is free; no custom backend).
- Biggest risk: Notion API breakage or permission model changing mid-sprint. Mitigation: lock API version early, build Asana as backup integration by Week 6 to prove extensibility.
- Recommendation: Build now, but validate Notion API integration first (Day 1, before full scaffolding). If OAuth works and API docs are solid, you have a 10-week sprint. If APIs are flaky, pivot to Asana-first or reconsider scope.

Snapshot

Metric	Value
Total Timeline (MVP -> Chrome Web Store Launch)	8-10 weeks
Recommended Team Size	1 full-time engineer + 1 product/design (can be founder)
Total Budget Envelope	$16K-$24K
Development Methodology	Kanban (continuous flow, no fixed sprints; extensions have shorter iteration cycles)
First Shippable Milestone	Week 4 -- internal alpha with Notion + 3 templates
Public Launch Target	Week 9 (Chrome Web Store submission)
Post-Launch Focus	Asana + 1 additional integration (ClickUp or Todoist), metrics dashboarding

Project Overview

QuickCreate is a client-heavy, server-light browser extension -- minimal infrastructure, fast feedback loops, clear MVP boundary.

Estimated Total Duration: 8-10 weeks from project kickoff to Chrome Web Store live.

Team Size Recommendation:

- Engineer (1 FTE, 8-10 weeks): Responsible for architecture, popup UI, Notion API integration, Manifest v3 setup, Chrome Web Store packaging, and deployment.
- Product/Design (0.5-1 FTE, 8-10 weeks, can be founder): User flows, popup UX, settings page, launch messaging, onboarding docs, pilot user recruitment.
- QA/Launch Support (0.25 FTE, Weeks 7-10): Smoke testing, bug triage, store listing prep.

Total headcount: 1.5-2 people. A solo founder with decent TypeScript chops can ship this in 12-14 weeks; adding a product/design person compresses to 8-10.

Development Methodology: Kanban (not Scrum).

Why not Scrum? Browser extensions ship smaller, more frequent updates. Fixed 2-week sprints create artificial batching. Instead: continuous flow where features move from Backlog -> In Progress -> Review -> Done as they're ready. Daily 15-minute standups, weekly milestone checks.

Phase Breakdown

Phase 1: Foundation & Setup (Week 1-2)

Get the project scaffolding, auth, and local data layer working before touching UI.

Week 1 Goals:

1. Validate Notion API integration pathway -- confirm OAuth 2.0 works, API rate limits, and permission scopes needed.
2. Project scaffolding using **Plasmo Framework** (TypeScript + React + Manifest v3 out-of-box).
3. Chrome Storage API schema design (user settings, OAuth tokens, templates, usage logs).
4. CI/CD pipeline (GitHub Actions -> Chrome Web Store auto-publish on tag).

Week 1 Tasks:

- Clone Plasmo starter template; set up repo with TypeScript strict mode.
- Build Chrome Storage API wrapper (get/set/remove with type safety).
- Implement OAuth 2.0 redirect flow for Notion (no UI yet -- CLI test only).
- Create `.env` template for Notion client ID / secret; add to GitHub Secrets.
- Set up GitHub Actions workflow: lint -> build -> upload to Chrome Web Store (no auto-publish yet; manual approval first).

Week 2 Goals:

1. Working OAuth for Notion (redirect -> token storage in Chrome Storage -> authenticated API calls).
2. Basic popup skeleton (no styling yet -- just wireframe).
3. Settings page to configure default Notion workspace/database.

Week 2 Tasks:

- Implement Notion OAuth callback handler; test token refresh flow.
- Hardcode one test template ("Research Task"); verify end-to-end task creation in Notion sandbox workspace.
- Build popup.tsx with three buttons (hard-coded templates); no styles yet.
- Build options.tsx (settings page) to select workspace and database from Notion.
- Add error logging (Chrome console + simple error toast).

Dependencies:

- Notion API documentation (check rate limits, required scopes: `read`, `write` on databases).
- GitHub Personal Access Token for Actions.

Estimated Effort: 10-12 person-days.

Deliverables:

- Plasmo project scaffolding with working build pipeline.
- Chrome Storage API wrapper with TypeScript types.
- Notion OAuth 2.0 integration (login -> token storage -> API access).
- Internal alpha: can create a hardcoded "Research Task" in Notion from extension popup.

Phase 2: Core MVP Features (Week 3-6)

Sprint 2A: Template System & Context Capture (Week 3-4)

Goal: Ship 3 templates ("Research Task," "Meeting Follow-up," "Bug Report") with auto-captured page context (URL, title, selected text).

Tasks:

- Define template schema: `{ id, name, description, platform, fields: [{ name, value, type, required }], created_at }`.
- Build template registry (hardcode 3 templates into extension; later allow custom ones).
- Implement context capture:
  - Page URL and title via `chrome.tabs.query()`.
  - Selected text via content script message passing.
  - Timestamp via JavaScript `Date.now()`.
- Build popup UI to select template -> auto-fill fields -> "Create Task" button.
- Test each template creates correct task structure in Notion.

Key Decision: **Hardcode templates vs. allow custom creation in v1? Answer: Hardcode.** Custom template builder adds 2 weeks. Ship with 3 battle-tested templates; custom creation is premium

feature (post-launch).

Dependencies:

- Content script to capture selected text (requires background service worker coordination in Manifest v3).

Estimated Effort: 8-10 person-days.

Deliverables:

- 3 templates fully mapped to Notion database properties.
- Context capture working (URL, title, selected text auto-filled).
- Popup UX: select template -> review fields -> create.

Sprint 2B: UI Polish, Error Handling & Settings (Week 5-6)

Goal: Make the extension feel finished; handle failures gracefully; allow users to configure their workspace.

Tasks:

- UI Polish:
  - Implement Tailwind CSS (lightweight, no build overhead).
  - Popup design: 400px wide, clean template selector, field review form, success/error feedback.
  - Settings page: workspace/database picker (fetch from Notion API), test connection button.
  - Light/dark mode toggle (use `prefers-color-scheme`).
- Error Handling:
  - Network errors -> "Check your internet" toast.
  - Invalid Notion token -> "Reconnect your Notion" + redirect to settings.
  - Missing database -> "Configure your workspace" link.
  - Rate limit hits -> "Please wait before creating another task."
- Analytics (local, no PII):
  - Track: templates used, tasks created, errors encountered (stored in Chrome Storage, not sent anywhere yet).
  - Show in popup: "You've created 12 tasks this week."

Dependencies:

- Notion API docs for error codes and rate limit headers.

Estimated Effort: 8-10 person-days.

Deliverables:

- Polished, usable popup with visual hierarchy and error feedback.
- Settings page with workspace configuration.
- Local usage analytics dashboard.

Phase 2 Summary:

- Weeks 3-6 Total Effort: 16-20 person-days.
- Exit Criteria: Internal alpha testers (5-10 people) can create 5+ tasks per day without friction.

Phase 3: Integration & Multi-Platform Prep (Week 7-8)

Don't add Asana to MVP, but build the integration framework so adding it takes 3 days, not 3 weeks.

Goal: Prove extensibility; prepare infrastructure for second integration; optimize for launch.

Week 7 Tasks:

- Refactor template system to be platform-agnostic:
  - Instead of hardcoding Notion field mappings, define platform adapters (`notion.adapter.ts`, `asana.adapter.ts`).
  - Each adapter translates template fields -> platform API format.
  - This costs 2 days now; saves 5 days when adding Asana.
- Performance optimization:
  - Audit popup load time (target: <500ms from click to visible).
  - Lazy-load settings page (only fetch Notion workspaces when user opens settings, not on startup).
  - Cache workspace/database list in Chrome Storage with 1-hour TTL.
- Security review:
  - Ensure OAuth tokens are stored in `storage.local` (encrypted by Chrome), never logged or sent to external services.
  - Add `Content-Security-Policy` header to prevent XSS.
  - Test malicious input (long URLs, special characters) doesn't break task creation.

Week 8 Tasks:

- Asana integration skeleton (do NOT ship yet; just verify architecture):
  - Build Asana OAuth handler (copy from Notion, swap client ID/secret).
  - Verify Asana API task creation endpoint works with test token.
  - Add Asana adapter skeleton; **do not** integrate into UI.
  - This proves: "If we want to add Asana, we can do it in 3 days," which is a major sellable feature to power users.
- Documentation for maintainability:
  - README: setup, build, test, deploy.
  - ARCHITECTURE.md: folder structure, data flow, platform adapter pattern.
  - CONTRIBUTING.md: how to add a new integration.

Dependencies:

- Asana API docs.
- Performance profiling tools (Chrome DevTools, Lighthouse).

Estimated Effort: 8-10 person-days.

Deliverables:

- Platform-agnostic template system (future integrations 3x faster to add).
- <500ms popup load time verified.
- Asana integration skeleton (not in UI, just proven architecture).

Phase 4: Testing & QA (Week 9)

Compress into 1 week because the MVP surface is small.

Goals:

1. Zero critical bugs before Chrome Web Store submission.
2. 80%+ code coverage on core flows (OAuth, template selection, task creation).
3. Test against Notion sandbox environment (never risk production data).

Testing Plan:

Test Type	Scope	Effort	Owner
Unit Tests	OAuth flow, template rendering, context capture	2 days	Engineer
Integration Tests	End-to-end task creation (3 templates)	2 days	Engineer
Notion Sandbox Test	Create 20 tasks across all templates; verify fields	1 day	Engineer
User Acceptance (UAT)	5-10 pilot users create 50+ tasks; collect feedback	3 days	Product/Design
Chrome Web Store Policy Check	Verify extension complies with store guidelines	1 day	Product
Performance Profiling	Load time, memory usage, CPU during task creation	1 day	Engineer

Tools:

- Jest for unit/integration tests (works with Plasmo out-of-box).
- Playwright **or** Puppeteer for end-to-end automation (optional; UAT is more valuable here).
- Chrome DevTools for performance profiling.

UAT Logistics:

- Recruit 5-10 beta testers from: Twitter, ProductHunt, Reddit r/productivity (post "Looking for beta testers" on Week 6).
- Send Chrome extension ZIP + settings doc + feedback form (Google Form).
- Weekly async check-ins; no calls needed.

Estimated Effort: 6-8 person-days.

Deliverables:

- Jest test suite covering OAuth, template logic, Notion API calls (target: 75%+ coverage).

- UAT feedback doc (feature requests, bugs, UX friction).
- Chrome Web Store compliance checklist (privacy policy, permissions justified, etc.).

Phase 5: Launch Preparation & Submission (Week 10)

Final sprint to Chrome Web Store live.

Tasks:

1. Chrome Web Store Listing:
   - Icon (128x128px): clean, simple task-based logo.
   - Screenshots (1280x800px): 5 screenshots showing popup flow, settings, tasks created.
   - Video trailer (optional): 30-second demo of creating a task from any webpage.
   - Description: "Create tasks in Notion in 15 seconds--no app switching, just click and go."
   - Privacy policy: "QuickCreate stores your Notion auth token locally in your browser. No data is sent to our servers."
   - Keywords: "task management, productivity, Notion, templates, extension."
2. Build & Package:
   - Run `npm run build` with production env vars.
   - Generate signed ZIP for Chrome Web Store.
   - Test extension in incognito mode (privacy).
3. Store Submission:
   - Pay $5 one-time Chrome Web Store developer account fee (if not done already).
   - Upload ZIP, screenshots, description, privacy policy.
   - Select: "Category: Productivity," "Visibility: Public."
   - Expect 24-72 hour review; prepare for potential rejection (most likely: missing privacy policy or over-broad permissions).
4. Post-Approval Setup:
   - Configure GitHub Actions to auto-publish future releases to Chrome Web Store (optional, add post-launch).
   - Update repo README with "Install from Chrome Web Store" link.
   - Prepare launch tweet, ProductHunt post, Reddit thread.
5. Monitoring & Fallback:
   - Set up Sentry (free tier) for exception tracking post-launch (optional; can use Chrome console logs first).
   - Document how to pull extension if critical bug found (remove from Web Store, push fix, re-submit).

Estimated Effort: 3-4 person-days.

Deliverables:

- Chrome Web Store listing live.
- Extension publicly available and installable.

Resource Plan

Budget Range from Idea Context: Medium ($16K-$24K for MVP launch is reasonable).

Role	Allocation	Duration	Monthly Rate	Total Cost
Senior Engineer (TypeScript/React/Manifest v3)	1 FTE	10 weeks	$8K-$10K/mo	**$18K-$23K** (10 weeks = 2.5 months)
Product/Designer (UX, launch)	0.5 FTE	10 weeks	$4K-$6K/mo	**$5K-$7.5K** (part-time or founder role)
QA/Launch Support (optional, weeks 7-10)	0.25 FTE	4 weeks	$3K-$4K/mo	**$3K-$4K** (or skip; engineer does QA)
Notion API docs, GitHub, DevTools	Tools	10 weeks	Included	**$0**
Subtotal Labor	--	--	--	**$23K-$34K**

Infrastructure & Services (10-week MVP):

Service	Cost	Notes
GitHub (private repo)	Free	Open source is fine; private for proprietary features.
Chrome Web Store developer account	$5 one-time	Pay once, list forever.
Notion API (free tier)	Free	3 requests/sec, sufficient for MVP; 100K+ tasks/month.
Sentry or LogRocket (optional)	Free tier	Exception tracking post-launch.
Google Forms (UAT feedback)	Free	Collect pilot feedback.
Subtotal Services	**$5-$50**	Effectively free.

Third-Party Libraries (bundled in extension, $0 runtime cost):

- Plasmo, React, TypeScript, Tailwind CSS, axios -- all open source.

Revised Budget Breakdown:

Category	Estimated Cost
Engineering (10 weeks, 1 FTE @ $8.5K/mo avg)	$21K
Product/Design (10 weeks, 0.5 FTE @ $5K/mo avg, or founder)	$6K (or $0 if founder)
QA/Support (optional, 4 weeks, 0.25 FTE)	$3K (or skip)
Services & Tools	~$50
Contingency Buffer (10%)	$3K
TOTAL MVP TO LAUNCH	**$18K-$27K**

Recommendation: Hire a competent engineer ($8-10K/mo) for 10 weeks; founder handles product/design + recruiting beta testers. This is $20K-$23K, fits your medium budget.

If you're a capable engineer yourself, you can ship solo for $5K services + your sweat equity.

Risk & Mitigation Timeline

Risk	Impact	Likelihood	Mitigation	When to Address
Notion API changes or breaks mid-sprint	Weeks 1-2 OAuth, weeks 3-6 template mapping all fail.	Medium (APIs stable, but permissions change quarterly)	**Week 1:** Test OAuth end-to-end before starting templates. Lock Notion API version in code. Have Asana as fallback.	**Day 1-3** of project. If Notion doesn't work by end of Week 1, pivot to Asana-first or Todoist.
Chrome Manifest v3 incompatibilities surface during test	Extension won't load; Plasmo framework issue.	Low (Plasmo is stable, but Manifest v3 is new)	**Week 2:** Test extension build in Chrome dev mode every 3 days. Subscribe to Plasmo GitHub issues.	**Weeks 1-2.** Have fallback: Manifest v2 branch (older, less secure, but proof-of-concept).
OAuth token storage insecure (Chrome Security Review flags)	Extension rejected from Web Store. 1-week delay.	Medium (common rejection reason)	**Week 2:** Review Chrome Web Store security docs. Store tokens in `storage.local` (encrypted), never in memory or logs. Privacy policy pre-written Week 8.	**Week 2 and Week 8.** Run internal security audit Week 8 before submission.
Pilot UAT reveals major UX friction (e.g., template selection confusing)	Low adoption in beta; launch timing slips 1-2 weeks.	Medium (common with unfamiliar UI patterns)	**Week 6:** Recruit beta testers early; collect feedback by Week 8. Design quick wins (e.g., re-order templates, add icons).	**Week 8 feedback review.** Have 2-3 days of UX burn-down scheduled Week 9.
Notion workspace/database selector breaks for non-admin users	Extension only works if you own the Notion workspace. Kill 50% of TAM.	Medium (permission model is complex)	**Week 3-4:** Test with non-admin Notion user during template phase. Verify API returns correct perms. Document workaround (admin shares database).	**Week 4.** If unfixable, pivot: only support admin workspaces (narrower TAM but viable).
Team member leaves mid-sprint	Ramp time, knowledge loss, delay.	Low (depends on team health)	Hire senior engineer, not junior (less context-dependent). Pair-programming Week 1-2 (engineer + designer on architecture review). Document as you go.	**Week 0:** Hire for retention; onboard with architecture pair-session.
Chrome Web Store submission rejected for non-obvious policy violation	1-3 week delay; resubmission required.	Low-Medium (common gotchas: over-broad permissions, confusing privacy policy, auto-start)	**Week 8:** Audit: permissions list only `scripting` + `tabs` + `storage`. Write privacy policy. Check for any auto-startup code.	**Week 8.** Have draft listing reviewed by someone who's shipped to Web Store before.

Milestone Calendar

Milestone	Target Date	Success Criteria
Project Kickoff & Setup Complete	End of Week 1	Plasmo scaffold, GitHub Actions, Chrome Storage wrapper working. Notion OAuth path validated.
Core MVP Features Complete	End of Week 6	3 templates fully functional in Notion; context capture working; UI polished. Internal testing passes; zero critical bugs.
Asana Integration Skeleton & Docs	End of Week 8	Platform adapter pattern proven; Asana OAuth flow tested (not in UI). README + ARCHITECTURE docs written.
UAT & QA Complete	End of Week 9	5-10 beta testers used extension for 1+ week; feedback doc collected. Jest test suite at 75%+ coverage. Chrome Web Store compliance checklist signed off.
Chrome Web Store Live	End of Week 10	Extension public, installable, 50+ installs in first week (success signal for next phase).
Post-Launch (Month 2)	Week 13-14	Asana integration live (if 40%+ of users request it). Weekly active users tracked. First community feature request identified for v1.1.

Budget Estimate Summary

Total MVP to Launch:

Category	Low Estimate	High Estimate	Notes
Engineering Labor (10 weeks, 1 FTE @ $8-10K/mo)	$20K	$25K	Senior engineer, TypeScript + React + Manifest v3 expertise.
Product/Design Labor (10 weeks, 0.5 FTE @ $4-6K/mo)	$5K	$7.5K	Can be founder; UI/UX + beta recruitment.
QA/Support (optional, 4 weeks)	$0	$3K	Engineer does QA in-house unless you hire.
Third-Party Services (Chrome Web Store, GitHub, Notion)	$50	$50	One-time $5 Chrome fee, rest free.
Contingency (10% buffer for scope creep)	$2.5K	$3.5K	Delays, extra integrations, bug fixes.
TOTAL	**$27.5K**	**$39K**	

Recommended Lean Option (Founder + 1 Engineer):

- Founder: Product + Design + Launch (sweat equity, $0).

- Engineer: 10 weeks @ $9K/mo = $22.5K.
- Services: $50.
- Contingency: $2.5K.
- Total: $25K. Fits your medium budget perfectly.

Go/No-Go Gates:

1. If Notion API doesn't work by end of Week 1: Pivot to Asana-first or shelve project. Cost: $3K sunk (setup + learning); save $22K.
2. If UAT feedback shows >50% confusion on template selection: Simplify UI (add icons, reorder). 1 week delay acceptable. Cost: +$3K labor.
3. If Chrome Web Store rejects for >2 reasons: Fix and resubmit; 1 week delay. Cost: negligible.

Post-Launch Plan (Months 3-6)

Month 2: Stabilization & First Integration (Asana)

- Weeks 13-14: Ship Asana integration (platform adapter already built; 3-5 days of eng).
  - Validation: If 30%+ of users connect Asana, it was worth it. If <10%, pause integrations.
- Weeks 13-16: Community feedback loop.
  - Collect top 5 feature requests from Discord/Reddit.
  - Publish roadmap (transparency builds trust).
  - Example hot requests: ClickUp, custom templates, task labels, recurring templates.

Month 3: Metric-Driven Iteration

- Install & Activate:
  - Target: 500 installs by Week 16 (realistic for small extension, no paid marketing yet).
  - Weekly Active Users: Target 20%+ (100 out of 500 using 2+ times/week).
- Feature Validation Matrix:
  - Track which templates are used most (should be "Meeting Follow-up" for remote workers).
  - If "Bug Report" <10% usage, replace with user-requested template.
- Monetization Soft Launch:
  - By Week 16, ship "Pro" tier ($9/mo) gating custom templates + advanced integrations.
  - No aggressive upsell; just make it available.
  - Target: 5% conversion (5 paying users = $45/mo recurring; covers 1/3 of an engineer's salary). If 0%, don't pursue monetization; focus on growth.

Months 4-6: Scaling & Sustainability

- Integrations (if validated):
  - ClickUp integration (ClickUp has 1M+ users; high potential).
  - Todoist integration (SMB market, lower complexity).
  - Asana (if not done by Month 3).
  - Ship 1 integration/month; measure adoption.

- Growth Channels:
  - ProductHunt launch (Month 2, Week 13-14).
  - Subreddits: r/productivity, r/remotework, r/entrepreneur.
  - Twitter/X: productivity community (makers, indie hackers).
  - Email: weekly tips for power users (engaged segment, build list).
- Retention Levers:
  - 30-day usage report: "You saved 2 hours creating tasks with QuickCreate."
  - Notion integration deep-dive: database property auto-mapping.
  - Community Discord/Slack: early adopters + feature feedback.
- Team & Sustainability:
  - If 1K+ users by Month 4, hire Part-Time Community Manager ($2K/mo) to handle Discord + feedback.
  - Re-assess Monetization: if Pro conversion <2%, shift to sponsorships (Notion, Asana paying to feature their integration) or B2B licensing.
  - Evaluate: Keep shipping features, or shift focus to another idea? If DAU flatlines, kill it.

Metrics Dashboard (Track from Day 1 of Launch):

Metric	Month 2 Target	Month 4 Target	Month 6 Target
Total Installs	300	800	2K
Weekly Active Users	60 (20% DAU)	160 (20% DAU)	400 (20% DAU)
Tasks Created	5K	20K	50K
Pro Conversions	0-2	10-20	30-50
Churn Rate (30-day)	<50%	<40%	<35%
Primary Integrations Used	Notion 100%	Notion 70%, Asana 25%	Notion 60%, Asana 30%, ClickUp 10%

Decision Checklist

Complete these actions in the next 7-30 days to de-risk and start executing:

1. Validate Notion API integration (Days 1-3).
   - Create a Notion test workspace (free tier).
   - Register OAuth 2.0 app on Notion developers portal.
   - Write a minimal Node.js script to auth and create a task in a Notion database.
   - Success = if you can create a task via API in <30 minutes, proceed. If >2 hours or API is broken, pivot to Asana-first or kill project.
2. Recruit your engineer or confirm you'll code this yourself (Days 1-7).
   - If hiring: post "Senior TypeScript/React/Manifest v3 engineer, 10-week contract" on Upwork or AngelList.
   - Minimum bar: 5+ shipped browser extensions, Manifest v3 experience, can onboard in 2 days. Interview should take 30 min; first week should be unblocked (not

context-dependent on you explaining everything).

If solo: block 50-60 hrs/week for 10 weeks; line up a designer/product person for feedback (can be unpaid co-founder).

3. Write your Notion privacy policy (Days 7-14).

   Required for Chrome Web Store; 200 words max: "QuickCreate stores your Notion auth token encrypted in your Chrome browser. We do not send any data to external servers. You can disconnect Notion anytime."

   Why now? If your policy requires data collection you haven't built, you have a scope problem early. If policy is simple, it's a 30-min job done.

4. Sketch 3 hardcoded templates (Days 7-14).

   On paper or Figma: what fields does "Research Task" need? ("Title," "URL," "Notes"). What about "Meeting Follow-up"? ("Title," "Attendees," "Action Items," "Date").

   Why? Maps to Notion database properties; this informs whether your Notion API call is 2 lines or 20. If fields are complex, you'll know before building.

   Share with 2-3 target users (remote workers, freelancers): "Does this template match your workflow?" Confidence threshold: >70% find it useful.

5. Set up GitHub repo, Plasmo scaffold, and GitHub Actions (Days 14-21).

   `npm create plasmo --template=react` -> GitHub public repo.

   Add GitHub Actions workflow: linting + build on every push.

   Add `NOTION_CLIENT_ID`, `NOTION_CLIENT_SECRET` to GitHub Secrets.

   Confirm `npm run dev` launches extension in Chrome and you can test it locally.

   Deliverable: By Day 21, you have a working dev environment; zero features yet, but you can test as you build.

6. Recruit 5-10 beta testers for Week 8 UAT (Days 14-21).

   Post in: r/productivity, r/remotework, Twitter #ProductHunt, indie hacker Discord servers.

   Message: "Building a Notion task-creation extension. Want to beta test? 15 min setup, report bugs + feedback."

   Collect: Email, Notion workspace name, occupation (ensures you have remote workers + freelancers).

   Don't pay them yet. Early-adopter mindset = free feedback. If you need incentives, offer lifetime Pro access (post-launch) or $50 gift card.

7. Lock MVP scope in a shared doc (Days 21-28).

   Shared Google Doc or GitHub Markdown: "QuickCreate MVP = 3 templates, Notion-only, browser extension, freemium pricing (free/Pro at $9/mo), no custom templates in v1, no Asana/ClickUp in v1, no team features."

   Share with your engineer (if hiring) so there's no scope creep mid-sprint.

   Why? Founder scope creep ("Can we also add...") is the #1 extension project killer. Lock it now.

Sources & References

1. Notion API Documentation -- https://developers.notion.com

2. Asana API Documentation -- https://developers.asana.com

3. Plasmo Framework (Manifest v3) -- https://www.plasmo.com

4. Chrome Web Store Publishing Guidelines -- https://developer.chrome.com/docs/webstore/publish/

5. Chrome Storage API (local & sync) -- https://developer.chrome.com/docs/extensions/reference/storage/

6. Manifest v3 Migration Guide -- https://developer.chrome.com/docs/extensions/mv3/intro/

7. TypeScript + React Best Practices for Extensions -- https://www.plasmo.com/docs/quickstarts/browser-platform/web

8. GitHub Actions for CI/CD -- https://docs.github.com/en/actions

9. Tailwind CSS (lightweight styling) -- https://tailwindcss.com

10. Jest Testing Framework -- https://jestjs.io

11. --

12. Bottom Line: This is a buildable, focused project. You have 10 weeks to validate the core idea (task creation speed matters) and prove Notion-specific demand. If by Week 10 you've got 50+ active users and 20%+ weekly engagement, you have a sustainable business to iterate on. If adoption stalls, you'll know by Month 3, and you can pivot or kill without sinking more than $25K. Ship it.

Ready to Build This?

Turn this blueprint into working software.

Our team builds custom software for founders —
we'll take this blueprint and ship the product.

Contact us: **hello@unbuiltlab.com**

Unbuilt Lab

unbuiltlab.com