PRD for Project Pitch: DailyForge

Cooper Kane

21 September 2025

Abstract

DailyForge is a lightweight, science-informed habit-building web platform designed to help users “forge” lasting routines. It combines a simple habit tracker, a collaborative tips board, and a knowledge base of evidence-based strategies. The Minimal Viable Artifact (MVA) focuses on simplicity, consistency, and community reinforcement, drawing on established behavioral psychology research.

1. Introduction

1.1 Context:

While tools like Habitica, Notion templates, and Google Sheets provide habit tracking, they often introduce unnecessary complexity, excessive gamification, or lack meaningful social support. Users frequently abandon these tools due to friction or loss of motivation.

1.2 Vision:

DailyForge seeks to bridge the gap by creating a clean, intuitive, and evidence-backed platform that empowers users to form sustainable habits. The platform emphasizes consistency, visual feedback, and peer-shaped strategies, with a minimal design that avoids feature bloat.

1.3 Problem Statement:

Many individuals struggle to maintain consistent habits despite motivation and access to information. Current solutions fail to balance simplicity, scientific credibility, and community support.

1.4 Goals:

- Provide an easy-to-use habit tracker that reinforces daily consistency.
- Offer community-driven tips for shared strategies and accountability.
- Ground the system in psychological research for credibility and trust.

2. Objectives and Success Metrics

2.1 Objectives:

- Help users form and sustain positive habits through simple tracking, visualization, and community reinforcement.
- Provide a credible, evidence-based framework for habit formation that is accessible to everyday users.
- Support iterative design cycles that balances feasibility with user value.

2.2 Success Metrics:

- Adoption KPI: 50+ unique users actively testing the platform within the first month.
- Engagement KPI: $\geq$ 70% of sessions include at least one habit completion action.
- Community KPI: $\geq$ 30% of users interact with the Forge Tips Board in their first week.
- Retention KPI: $\geq$ 50% of users return within 7 days of initial use.
- OKR: By the end of first quarter after release, demonstrate that at least 6 users report creating a consistent habit with the help of DailyForge. This can be achieved through a post-use survey.

3. Scope

3.1 In-Scope:

- Habit Tracker Core:
    - Create Habits
    - Define Frequency and Duration
    - Mark Completion
    - View Streaks and Weekly Charts
- Forge Tips Board:
    - User Tips
    - Upvote Option
- The Anvil (Knowledge Base):
    - Static Cards with Science-based Facts
    - Citation of Sources

3.2 Out-of-Scope

- Reminders
- Advanced AI Habit Recommendations
- Forge Together - Group Habit Tracking
- Avatars
- Rewards
- Complex Achievement Systems

4. User Stories & Use Cases

4.1 Habit Tracking:

- "As a user, I want to check off my daily habit with one click so that I can see my progress without friction."
- "As a user, I want to see a streak counter so that I feel motivated to maintain consistency."

4.2 Forge Tips Board:

- "As a user, I want to read practical tips from other users so that I can learn effective strategies."
- "As a user, I want to upvote useful tips so that the best advice rises to the top."

4.3 The Anvil (Knowledge Base):

- "As a user, I want to have quick access to evidence-based strategies so that I know the science behind the app."

4.4 Edge Cases & Constraints:

- What if a user misses a day?
    - Streak resets or pauses (must define logic clearly).
- What if two users submit the same tip?

    - Tips may be similar; upvoting helps surface the better one.
- What if a user enters too many habits?
    - Risk of clutter; UI should remain clean.

5. Functional Requirements

5.1 Must Have Requirements (MVA):

- Habit Creation - Users can create habits with name, frequency, and duration. [1]
- Daily Check-in - Toggle completion for each habit with streak and chart visualization. [2] [3] [4]
- Forge Tips Board - Users can post short tips and upvote others. [5] [6]
- The Anvil (Knowledge Base) - Static cards with evidence-based strategies with source citations. [7] [8]
- Local Persistence - Prevention of data loss on refresh. [9]

5.2 Should Have Requirements:

- Moderation - Simple flagging system for spam, irrelevant tips, or profanity. [10]
- Responsive Design - Responsive design for both desktop and mobile browsers. [11]

5.3 Could Have Requirements:

- Reminders - Email or push notifications. [12]
- AI Tip Detection - AI-assisted duplicate tip detection. [13]
- Advanced Analytics - More advanced analytics for user progress. [14]

6. Non-Functional Requirements

6.1 Performance

- Dashboard loads in less than 2 seconds under normal usage. [15]

6.2 Scalability

- Support up to 10,000 concurrent users during initial launch. [16]

6.3 Accessability

- WCAG 2.1 AA compliance (keyboard navigation, screen reader support). [17]

6.4 Security & Privacy

- Encrypted user authentication, secure handling of habit data. [18]
- No collection of unnecessary personal information. [19]
- Personal data must not be shared with third parties. [20]

7. Dependencies & Risks

7.1 Technical Dependencies

- Programming Language: Python
- Backend Framework: Django
- Database: SQLite (development) or PostgreSQL (production-ready) via Django ORM.
- Frontend:
    - Option A: Django templating system
    - Option B: React or similar JS framework
- Development Environment: Visual Studio Code
- Version Control: Git + GitHub/GitLab
- Testing Tools: Djano's built-in test framework, Pytest (optional).

7.2 Third-Party Integrations:

- Django REST Framework (DRF): for creating APIs if React or mobile apps are considered later.
- Chart.js or Recharts (JavaScript Library): for streaks and progress visualization.
- Bootstrap or Tailwind CSS: for clean and responsive UI.
- Django-Allauth or Django's built-in Auth: if user authentication is required.
- SQLite/PostgreSQL: database engines accessed through Django ORM.
- Python Libraries (optional): pandas, profanity-check

7.3 Top Risks & Mitigations

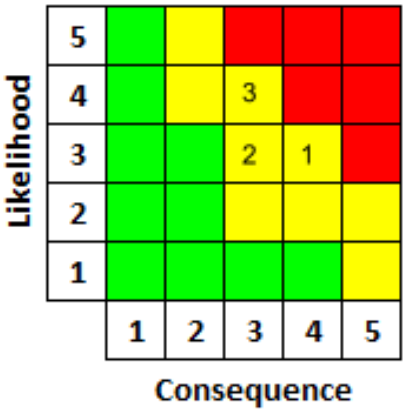

Figure 1. DailyForge Top 3 Risk Matrix

Risk Number	**Risk Title**	**Consequence (1-5)**	**Likelihood (1-5)**	**Mitigation**
1	Low User Engagement	4	3	Focus on clean user experience, progress visualization, and simple

				onboarding.
2	Spam or Irrelevant Tips	3	3	Community voting and flagging to filter low-quality content.
3	Scope Creep Beyond MVA	3	4	Strict feature freeze until after MVA release.

Table 1. DailyForge Top Risks & Mitigations

8. Acceptance Criteria

Requirement Number	Requirement Text	Verification Methodology	Verification Description
1	Users can create habits with name, frequency, and duration.	Test	Create habits with different names, frequencies, and durations; verify they appear correctly.
2	Users can mark habits as complete daily with a simple toggle.	Test	Mark habits complete each day; verify status updates correctly and persists.
3	The system displays a streak counter that increments correctly when habits are completed.	Demonstration	Complete habits in sequence and show that streak counter increments correctly and resets when missed.
4	Weekly progress is visualized in a chart (e.g., bar chart).	Demonstration	Enter a week's habit data and display the chart; show that it reflects actual completions.
5	Users can submit a short tip to the Forge Tips Board.	Test	Enter a week's habit data and display the chart; show that it reflects actual completions.
6	Users can upvote tips.	Test	Upvote tips and verify counts increment correctly; prevent duplicate votes.
7	The Anvil contains static cards showing evidence-based strategies.	Examine	Inspect the Anvil section to confirm all static cards are present and correct.

8	All knowledge base entries display citations in APA format.	Examine	Review knowledge base entries and confirm citations follow APA format.
9	Local persistence prevents data loss on refresh (habits and tips persist in local storage or database).	Test	Refresh the page or restart app; verify habits and tips persist.
10	The Forge Tips Board supports a simple flagging system for inappropriate tips.	Demonstration	Flag tips and show that flagged tips are marked or removed correctly.
11	The interface is responsive on desktop and mobile browsers.	Demonstration	Interact with interface on different devices; show that layout adjusts correctly.
12	Reminders are available via email or notification.	Demonstration	Trigger reminders and show that they are received correctly and timely.
13	AI-assisted duplicate tip detection.	Test	Submit duplicate tips; verify system detects duplicates and notifies the user.
14	Advanced analytics on user progress (graphs beyond streaks/weeklies).	Demonstration	Enter varied user progress data; display advanced graphs showing correct trends.
15	Dashboard loads in < 2 seconds (performance).	Analysis	Measure load times and confirm the dashboard meets <2 second requirement.
16	The system supports 10,000 concurrent users without degradation (scalability).	Analysis	Simulate load and review performance metrics to ensure the system remains stable.
17	The platform meets WCAG 2.1 AA accessibility standards.	Examine	Inspect accessibility features (labels, contrast, keyboard navigation) to verify compliance.

18	All data is securely handled, with encryption at rest and in transit.	Examine	Review security configurations to confirm encryption is applied at rest and in transit.
19	No unnecessary personal data is collected (privacy).	Examine	Review data collection procedures to confirm only required information is collected.

Table 2. DailyForge Requirement Acceptance Criteria

8.1 Verification Methods:

- Examine - Verification by visual inspection or manual review of artifacts (e.g. documents, code, configurations, or outputs). No execution is required.
    - General Example: Examining a UI design to ensure all required buttons and labels are present.
- Test - Verification by executing the system or component under controlled conditions with specific inputs, and checking that the actual outputs match the expected results.
    - General Example: Running a unit test to confirm that a login function rejects invalid credentials.
- Demonstration - Verification by showing that the system performs its intended function in an operational or simulated environment. Demonstrations are often interactive and focus on usability.
    - General Example: Demonstrating a search feature by typing in a keyword and showing the results page.
- Analysis - Verification through reviewing data, calculations, or performance metrics to confirm that requirements are met. Often involves evaluating logs, statistics, or benchmarks.
    - General Example: Analyzing server response time to confirm the system meets the requirement of <2 seconds per request.

AI Audit Log

Prompt	Accepted Output	Rejected Output	Revisions
Can you give me examples of a PRD introduction section?	I reviewed the examples. AI suggestions provided structural clarity.	The substance of the examples was not used.	Substance of the examples were revised to fit my project topic.
Can you show me examples of PRD scope sections?	I reviewed the examples. AI suggestions provided structural clarity.	The substance of the examples was not used.	Substance of the examples were revised to fit my project topic.
Can you name some examples of requirement verification methods?	Accepted: Examine, Test, Demonstration, Analysis.	Rejected: Inspection, Simulation, and Demonstrated Use/Field Test.	Revised the definitions under the accepted methods and deleted the rejected methods.
Can you review these to make sure I captured all the requirements in the table?	There were a few items that were identified to be requirements.	A few items were new additions to the document that was deleted. There were also some items selected that were not requirements.	Used mainly as a sanity check to ensure I didn't have any obvious requirements not listed.

Note: ChatGPT-5 was used for all searches