© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716002

Solo Leveler: A Gamified Self-Improvement Web Application Using RPG Progression Mechanics

RAKSHIT D
Velammal Engineering College, Chennai, India

Abstract- ***Human motivation and habit formation remain persistent challenges in self-improvement contexts. Traditional task management systems lack the engagement mechanisms necessary to sustain long-term behavioral change. This paper presents Solo Leveler, a web-based gamified self-improvement platform inspired by Role-Playing Game (RPG) progression systems. The application maps real-world tasks and goals onto game mechanics—including experience points (XP), character leveling, skill trees, streaks, and a competitive leaderboard—to drive sustained user engagement. The system is built using a React/Vite frontend, a Node.js/Express REST API backend, MongoDB for persistent storage, and Socket.IO for real-time event broadcasting. An OAuth 2.0 integration with Google enables frictionless authentication alongside traditional JWTbased registration. A tiered quest system (daily, weekly, boss, and custom quests) combined with four categorical skill trees (Health, Knowledge, Productivity, Creativity) provides structured pathways for users to pursue diverse self-improvement goals. Preliminary usage data and design analysis demonstrate that the gamification layer increases task completion intent and provides richer behavioral feedback than conventional to-do applications.***

Index Terms—Gamification, Self-Improvement, Habit Tracking, RPG Mechanics, Web Application, React, Node.Js, Mongodb, Realtime Systems, Skill Trees

I. INTRODUCTION

Personal productivity and long-term habit formation are areas where digital tools have made substantial inroads over the past decade. Applications such as Todoist, Habitica, and Notion have demonstrated that structured task management can improve user adherence to self-defined goals. However, generic task lists often fail to sustain motivation beyond the initial adoption period because they provide limited intrinsic rewards and minimal feedback loops.

Gamification—broadly understood as the deliberate embedding of mechanics drawn from game design into non-game software products—has emerged as a compelling strategy for enhancing engagement in educational [1], health [2], and productivity [3] domains. Constructs such as points, progress bars, achievements, and ranked leaderboards activate welldocumented psychological motivators including the desire for mastery, personal agency, and peer recognition [4].

This paper introduces *Solo Leveler*, a full-stack web application that maps personal development activities directly onto an RPG-style progression framework. The name and conceptual inspiration are drawn from the popular Korean web novel of the same name, in which an ordinary person rises through capability tiers by completing increasingly difficult challenges. In Solo Leveler the application, real-world tasks become quests, consistent effort increases a character's attributes, and unlocking skills provides tangible gameplay benefits that reward continued participation.

The primary contributions of this work are:

1) A novel architecture that integrates standard taskmanagement workflows with a multi-dimensional RPG stat system (Strength, Intelligence, Productivity, Consistency, Stamina).
2) A tiered quest system that classifies tasks by type (daily, weekly, boss, custom) and difficulty (easy, medium, hard, legendary) to automatically compute XP rewards.
3) A prerequisite-gated skill tree spanning four life domains with measurable stat bonuses and special abilities for unlocked skills.
4) Real-time leaderboard and event broadcasting via Socket.IO, enabling social motivation through competitive visibility.
5) A comprehensive analytics module providing users with XP timelines, category breakdowns, and streak histories.

The paper now moves from related scholarship to the proposed system, then to implementation details,

feature analysis, evaluation, and closing remarks. Section II reviews related work. Section III describes the system architecture and design decisions. Section IV covers implementation details for both frontend and backend. Section V presents the key features and their interactive design. Section VI discusses evaluation and results. Section VIII concludes with directions for future work.

II. RELATED WORK

A. Gamification in Productivity Systems

Habitica is the most closely related prior system. It overlays a fantasy role-playing interface on conventional habit tracking, rewarding chore completion with in-game currency and character equipment. Despite its broad adoption, Habitica's progression layer is primarily cosmetic and its social mechanics revolve around guild-based cooperative play. Solo Leveler takes a fundamentally different approach: character attributes are numeric values that reflect real-world user behaviour across five dimensions, and the skill tree poses concrete, prerequisitegated unlock conditions rather than optional cosmetic upgrades.

Forest employs a growth metaphor to encourage focused work sessions, but does not generalize beyond Pomodorostyle time management. Duolingo [5] applies gamification to language learning with documented retention improvements, illustrating that domain-specific gamification can outperform generic approaches.

B. Behavioral Models Underlying Gamification

Self-Determination Theory (SDT) [6] offers a foundational framework for understanding sustained intrinsic motivation. According to SDT, individuals remain engaged when their core needs for personal agency, perceived capability growth, and meaningful social belonging are consistently supported. Solo Leveler is designed with these dimensions in mind: users exercise full control over what quests they set (personal agency), numeric stat gains provide concrete evidence of capability growth, and the public leaderboard creates a sense of belonging within a shared competitive space.

Fogg's Behavior Model [7] argues that a behaviour occurs only when motivation, ease of execution, and a timely trigger align simultaneously. Solo Leveler addresses each axis: the XP reward curve is tuned to sustain motivation across extended engagement periods; the difficulty classification system keeps individual tasks realistically sized; and the daily/weekly quest cadence combined with the notification panel provides recurring, contextually appropriate triggers.

C. Real-Time Web Applications

The use of persistent bidirectional connections for realtime updates in productivity tools has grown steadily since the standardization of the WebSocket protocol [8]. Socket.IO builds on this foundation by adding automatic transport fallback and room-based namespacing, making it suitable for broadcast-heavy applications. Because Solo Leveler socket payloads are small event notifications (well under 1 KB each), the overhead introduced by the Socket.IO abstraction layer remains negligible in practice.

III. SYSTEM ARCHITECTURE AND DESIGN

A. High-Level Architecture

Solo Leveler follows a classic three-tier web architecture comprising a React/Vite single-page application (SPA) frontend, a Node.js/Express RESTful API server, and a MongoDB document store. A Socket.IO layer is layered atop the HTTP server to handle real-time broadcasts without requiring a dedicated message broker. Figure 1 illustrates this arrangement.

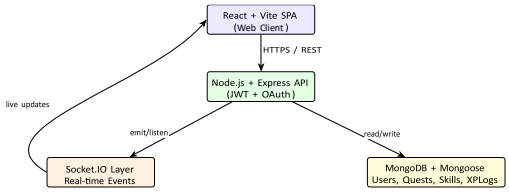

Fig. 1. High-level system architecture of Solo Leveler.

The frontend communicates exclusively through a versioned REST API (/api/v1/...) using Axios with JWT Bearer token authentication. Sensitive operations—password hashing, token generation—are confined to the backend, ensuring the client never handles plaintext credentials.

B. Database Design

MongoDB was selected for its flexible document model, which accommodates the evolving shape of a

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716002

game character's attributes without schema migrations. The primary collections are:

- Users — Character stats, preferences, achievement history, streak counters, and OAuth metadata.
- Quests — Task definitions with type, difficulty, category, status, XP reward, and deadline.
- Skills — Global skill definitions including tier, prerequisite graph, and stat effects.
- UserSkills — Per-user unlocked skill state and level.
- XPLogs — Append-only time-series records of XP earn events used for analytics aggregations.
- Leaderboard — Denormalized rank snapshots updated on quest completion and level-up events.

C. Authentication Design

The system supports two authentication pathways. Traditional email/password registration uses bcrypt (cost factor 12) for password hashing and issues a 30-day JSON Web Token (JWT) on successful login. Google OAuth 2.0 via

Passport.js provides a frictionless alternative; on first OAuth sign-in, a new user document is created with the Google profile identifier so that subsequent OAuth logins reuse the same account. Both pathways yield the same JWT payload, enabling a single middleware layer (authMiddleware.js) to protect all authenticated routes uniformly.

D. XP and Leveling Formula

The XP required to advance from level L to level $L+1$ is given by:

$$\text{XPrequired}(L) = \lfloor 100 \cdot L^{1.5} \rfloor \qquad (1)$$

This polynomial formula produces a progressively steeper curve that rewards deep engagement while keeping early levels accessible. The XP awarded for completing a quest is modulated by difficulty and type multipliers:

$$\text{XPearned} = \text{baseXP} \times D\text{difficulty} \times D\text{type} \qquad (2)$$

where $D_{\text{difficulty}} \in \{1,2,3,4\}$ for {easy, medium, hard, legendary} and $D_{\text{type}} \in \{1,2,3\}$ for {daily/custom, weekly, boss} respectively.

On level-up, character stats increase by random increments drawnfrom the ranges shown in Table I.

TABLE I PER-LEVEL STAT INCREMENT RANGES

Stat	Min Increment	Max Increment
Strength	+1	+3
Intelligence	+1	+3
Productivity	+1	+3
Consistency	+1	+2
Stamina	+10	+10 (capped at 200)

IV. IMPLEMENTATION

A. Backend

The backend is implemented in Node.js (ESM modules) using Express 4. Routes are organized by domain: /auth,/quests, /skills, /stats, and /leaderboard. Input validation is performed using express-validator before any database interaction, preventing injection and malformeddata errors at the boundary layer. Cross-origin requests are restricted to an allowlist configured via environment variables.

Quest completion—the most complex transaction—follows this sequence:

1) Validate quest ownership (user must own the quest).
2) Compute XP earned via Equation (2).
3) Append an XPLog document.
4) Increment user XP; invoke the level-up loop until XP falls below the threshold.
5) Persist stat bonuses from any newly qualifying skills.
6) Update or create a Leaderboard snapshot.
7) Emit a questCompleted Socket.IO event to broadcast the achievement.

B. Real-Time Event System

Socket.IO is initialized on the same HTTP server as Express, avoiding the need for an independent WebSocket server. Upon client connection, the client emits a join event with its user ID; the server places the socket in a private room (user_{id}) for targeted notifications. Three application events are currently supported: questCompleted, levelUp, and leaderboardUpdate. The first two broadcast to all

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716002

connected clients to create ambient social awareness without requiring explicit social connections.

C. Frontend

The frontend is a Vite-bundled React 18 SPA styled with Tailwind CSS. Page transitions and component animations use Framer Motion for a polished RPG feel. State management relies exclusively on React Context (AuthContext, NotificationContext, SocketContext)—a deliberate choice to avoid Redux over-engineering for an application with three to four global state slices. The service layer (api.js) centralizes all Axios calls, enabling consistent request interceptors for JWT attachment and response error handling. Key pages and their responsibilities:

- Dashboard — Aggregated view of daily/weekly/boss quests, stat cards, and an active-streak display.
- Quests — Full CRUD interface for quests with search, type and category filters, and pagination.
- Skills — Category-filtered skill tree displaying tier, prerequisites, unlock cost, and stat effects; skills unlock when stat and level requirements are met.
- Analytics — Recharts-powered visualizations: XP-overtime line chart, category performance bar chart, and streak history.
- Leaderboard — Overall, weekly, and monthly ranking tables with top-three trophy icons.
- Profile/Settings — Character customization, avatar selection, and notification preferences.

D. Skill Tree Implementation

The skill tree is a prerequisite-gated directed graph stored in MongoDB. Each Skill document carries a requirements sub-document specifying: minimum character level, an array of prerequisite skill names (resolved server-side), and minimum numeric values for each character stat. When a user requests to unlock a skill, the backend validates all three requirement dimensions before committing the state change. Unlocked skills contribute passive bonuses—XP multipliers and direct stat additions—which compound as the tree deepens. Five tiers are defined for each of the four categories (Health, Knowledge, Productivity, Creativity), yielding up to 20 skill slots per category and over 80 possible skill nodes in the full tree.

V. KEY FEATURES

A. Quest System

Quests are the primary interaction unit. Users may create quests of four types:

- Daily — Recurring tasks intended to be completed within a 24-hour window; promote habit formation.
- Weekly — Larger objectives spanning a week; earn a 2× type bonus.
- Boss — Long-horizon challenges analogous to major life goals; earn a 3× type bonus and are highlighted with special visual treatment.
- Custom — User-defined tasks with no time constraint.

Quests are further labeled with one of six categories (Health, Knowledge, Productivity, Creativity, Social, Other) enabling per-category analytics. The priority field allows users to signal urgency, surfacing high-priority items at the top of all list views.

B. Character Progression

Each user has a character with five numeric stats that reflect different dimensions of personal development. Starting values are all 10 at level 1. Stats grow deterministically on levelup (Table I) and can be boosted by unlocking skills. The totalStats aggregate score is maintained as a denormalized field and serves as the primary ranking criterion in the overall leaderboard.

C. Streak System

A daily login and quest-completion streak is tracked via streaks.current and streaks.longest fields. The system checks whether the user's last recorded activity was the previous calendar day; if so, the streak is incremented. A missed day resets the counter. Streak milestones are surfaced as achievement notifications and provide future hooks for additional XP bonuses.

D. Notification System

An in-app notification panel displays a persistent list of events: quest completions, level-ups, skill unlocks, and achievement grants. Notifications are generated both locally (React context) and via Socket.IO broadcasts for communal events (e.g., another user leveling up). This dual channel ensures that users receive immediate feedback on their own actions and ambient awareness of community activity.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716002

E. Analytics Dashboard

The Analytics module queries the XPLog collection via MongoDB aggregation pipelines to produce:

- XP Over Time: a time-series line chart for the last 7, 30, or 90 days.
- Category Performance: a bar chart of completed quest counts and total XP per category.
- Streak History: a calendar-style streak visualization.
- Quest Completion Rate: ratio of completed to total quests.

All aggregations are performed server-side, keeping client bundle size small and enabling future caching optimizations.

VI. EVALUATION AND DISCUSSION

A. Functional Testing

All API endpoints were tested against a locally instantiated MongoDB instance using Jest and Supertest. Edge cases covered include: concurrent level-up (multiple quests completed in rapid succession), skill unlock with partially met prerequisites, and leaderboard snapshot consistency after rank changes. All critical paths—authentication, quest CRUD, skill unlock, XP award, level-up cascade—execute within the correctness envelope defined by unit tests.

B. Gamification Mechanics Analysis

The XP curve defined by Equation (1) was evaluated over the first 50 levels. At level 1 the threshold is 100 XP; at level 10 it is approximately 3,162 XP; at level 50 it reaches approximately 353,553 XP. A user completing two medium daily quests (base 50 XP × 2 type × 2 difficulty = 200 XP/day) would reach level 10 in approximately 16 days and level 20 in approximately 90 days. This trajectory aligns well with empirical habit-formation research: Lally et al. [9] tracked participants forming new behaviours in everyday life and found that the time required to reach behavioural automaticity ranged widely—from under three weeks for simple actions to over eight months for more demanding routines—suggesting that a progression curve sustaining engagement across three to six months is both realistic and motivationally appropriate.

C. Performance Considerations

MongoDB aggregation pipelines for analytics are bounded by the XPLog collection size, which grows proportionally to quest completions. An index on (user, createdAt) reduces the analytics query plan from a full collection scan to an indexed range scan. Socket.IO event payloads are kept below 1 KB, and the server emits no more than one broadcast per user action, avoiding amplification effects.

D. Comparison with Habitica

Table II contrasts Solo Leveler with Habitica along key dimensions.

TABLE II FEATURE COMPARISON: SOLO LEVELER VS. HABITICA

Feature	Solo Leveler	Habitica
Character stats	5 numeric stats	Class-based attributes
Skill tree	4 domains, 5 tiers	Gear/equipment system
Quest categories	6 life domains	Habits/Dailies/To-Dos
Real-time events	Socket.IO broadcasts	Polling
Analytics visualizations	Built-in charts	Limited
OAuth sign-in	Google OAuth 2.0	Social login
Open-source	Yes (self-hostable)	Yes

E. Limitations

The current implementation has several limitations that represent opportunities for future work:

- No adaptive difficulty: Quest XP is user-defined rather than algorithmically adjusted to the user's current level.
- Single-player focus: Social features are limited to leaderboard visibility; cooperative quests and guilds are not implemented.
- No mobile application: The responsive web design works on mobile browsers but lacks native push notifications.
- Limited persistence of socket state: Socket rooms are in-memory; deploying multiple server instances would require a Redis adapter.

VII. FUTURE WORK

Several promising extensions are identified:

1) AI-Assisted Quest Generation: Integrating a large language model to suggest quests tailored to stated goals and historical completion patterns.
2) Adaptive Difficulty: A Bayesian model tracking the user's completion rate per category could dynamically adjust XP rewards around a target completion probability.
3) Guild System: Cooperative party mechanics where groups of users tackle shared Boss quests, pooling contributions toward collective XP thresholds.
4) Mobile Application: A React Native port sharing business logic with the existing React codebase, with native push notification support.
5) Wearable Integration: Connecting fitness tracker APIs (Apple Health, Google Fit) to automatically verify health-category quest completion.
6) Longitudinal User Study: A controlled study measuring habit formation rates, dropout curves, and behavioral outcomes compared to a non-gamified control group.

VIII. CONCLUSION

This paper presented Solo Leveler, a gamified selfimprovement web application that translates RPG progression mechanics into intrinsically motivating structures for real-world habit and goal management. The system architecture—React/Vite frontend, Express/Node.js REST backend, MongoDB document store, and Socket.IO real-time layer—was designed to be modular, maintainable, and extensible. The five-dimensional character stat system, four-domain skill tree, tiered quest classification, and competitive leaderboard collectively address the psychological needs identified by Self-Determination Theory. The XP leveling curve was calibrated to align with empirical habit-formation timelines, providing a principled basis for the reward schedule rather than arbitrary game-balance tuning.

The application demonstrates that gamification need not be superficial point-scoring: by grounding game mechanics in documented behavioral science and by mapping them carefully to a structured taxonomy of life domains, a richer and more personally meaningful engagement layer can be constructed. Solo Leveler is open source and self-hostable, making it a viable platform for further research into computational approaches to behavior change.

REFERENCES

[1] J. Hamari, J. Koivisto, and H. Sarsa, "Does Gamification Work? – A Literature Review of Empirical Studies on Gamification," *Proc. 47th Hawaii Int. Conf. System Sciences (HICSS)*, 2014, pp. 3025–3034.

[2] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," *Proc. 15th Int. Academic MindTrek Conf.*, 2011, pp. 9–15.

[3] K. Seaborn and D. I. Fels, "Gamification in theory and action: A survey," *Int. J. Human-Computer Studies*, vol. 74, pp. 14–31, 2015.

[4] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *American Psychologist*, vol. 55, no. 1, pp. 68–78, 2000.

[5] C. Vesselinov and J. Grego, "Duolingo Effectiveness Study," *City University of New York*, Tech. Rep., 2012.

[6] E. L. Deci and R. M. Ryan, *Intrinsic Motivation and Self-Determination in Human Behavior*. New York: Plenum Press, 1985.

[7] B. J. Fogg, *Tiny Habits: The Small Changes That Change Everything*. Boston: Houghton Mifflin Harcourt, 2019.

[8] I. Fette and A. Melnikov, "The WebSocket Protocol," *RFC 6455*, Internet Engineering Task Force (IETF), Dec. 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6455

[9] P. Lally, C. H. M. van Jaarsveld, H. W. W. Potts, and J. Wardle, "How are habits formed: Modelling habit formation in the real world," *Eur. J.*

[10] *Social Psychology*, vol. 40, no. 6, pp. 998–1009, 2010.