

Gamified Habit Tracker

Students:

- Bryan A. Fernandez
- Abhiram Bhogi
- Juan Lao Romero
- Shamar Weekes
- George Ulloa

Instructor/Faculty:

- Masoud Sadjadi, Florida International University



Problem Definition & Motivation

The Core Problem:

- The main challenge is the disconnect between aspiration and action in personal development and habit formation.
- Traditional self-help tools and to-do lists often lack a compelling feedback loop and a sense of progression.
- Users struggle with sustaining motivation because the rewards for incremental progress are intangible or too far in the future.
- Current solutions fail to integrate the cognitive engagement found in games with the necessity of real-world productivity.

Our Motivation: The Solution

- We aimed to gamify the journey of personal growth, transforming real-life tasks into elements of an engaging Role-Playing Game (RPG).
- **Bridging the Gap:** Completing real-world tasks (e.g., studying, exercising) translates into tangible in-game rewards.
- **Leveraging AI for Personalization:** AI generates unique, personalized quests and challenges, making the habit-formation process novel and tailored.
- **Fostering Long-Term Engagement:** Replacing transient checklist motivation with the long-term commitment characteristic of an RPG, encouraging sustained behavior change.
- **Impactful Metrics:** Success is defined by sustained adherence to habits, not just short-term task completion.



System Architecture & Design Overview

Client–Server Architecture: Decoupled React frontend communicates with a Django REST backend via secure HTTP/REST APIs.

Scalable Data Layer: SQLite relational database with indexed foreign keys, unique constraints, and ORM-based data access.

Modular Backend Design: Separated services for authentication, game logic, AI integration, and data persistence to ensure maintainability.

Component-Based Frontend: Reusable React components, custom hooks for state management, and Tailwind CSS for responsive UI design.



Implementation Highlights & System Demonstration

Implementation Highlights:

- **Full-stack architecture:** React + Vite frontend, Django REST backend, SQLite database, secured with JWT authentication.
- Centralized XP and leveling engine with AI-powered quest difficulty (GPT-4o-mini) plus deterministic fallback when AI is unavailable.
- **Rich game systems:** five attributes, streaks and bonuses, achievements, daily check-ins, character/appearance/theme unlocks, and Tower mode.
- **Robust quality & integrity:** database constraints for no duplicate active quests, automatic stat updates via signals, and unit/integration/API tests.

System Demonstration:

- **New user journey:** register, log in, and show the initial dashboard (character, level 1, empty quest list).
- **Create quests:** add an AI-analyzed custom quest, then import preset quests to show batch import and duplicate-quest prevention.
- **Complete a quest:** trigger completion, display the celebration modal with XP breakdown, and demonstrate level-up plus new character unlock.
- **Engagement features:** perform a daily check-in, briefly showcase Achievements, and start a Tower floor to highlight long-term progression.



Testing, Evaluation, & Results



Executed multiple testing strategies for reliability and performance.



Conducted Unit, Integration, API, and User Acceptance Tests.



Results confirmed correct XP logic and character progression.



Testing identified and fixed logical inconsistencies and UI bugs.

Challenges, Lessons Learned, & Future Work

Challenges Overcome

- **AI API Cost Mitigation:** The high financial risk of AI API costs required proactive usage limits from the outset.
- **Data Integrity Enforcement:** Preventing duplicate quest creation demanded synchronized enforcement across both the UI and database layers.
- **XP Consistency & Balance:** Achieving the optimal balance in XP calculation to maintain a system that is challenging yet consistently rewarding.
- **Auth System Implementation:** Building a **robust and scalable user authentication system** presented initial integration complexity.

Lessons Learned

- AI usage was **strictly limited to user-created quests** to mitigate high API costs
- Duplicate quest prevention was enforced using **unique constraints at the database level**.
- A **centralized XP logic** and thorough unit testing ensured calculation consistency.
- **JWT-based authentication** was selected for enhanced security and scalability over session-based methods

Future Work

- **Deepen Social Engagement:** Introduce a friend system, achievement sharing, and collaborative challenges.
- **Advanced Analytics:** Develop dashboards, heatmaps, and AI-driven habit recommendations.
- **Mobile Experience:** Create a **React Native application** for push notifications and offline access.