

Computer Science - Capstone Project Documentation

Knight Foundation School of Computing and Information Sciences (KFSCIS) — Florida International University

Instructor: Masoud Sadjadi | Version: 2025 Fall | License: MIT (include in your repo)
Name: Bryan A. Fernandez, Abhiram Bhogi, Juan Lao Romero, Shamar Weekes, George Ulloa

ACM Student Outcomes (O1–O6)

Outcome	Description
O1. Problem Analysis & Requirement	Elicit and analyze user/stakeholder needs; define constraints and success criteria
O2. System Design & Implementation	Design solutions using appropriate abstractions, patterns, and tools; implement maintainable code.
O3. Experimentation, Testing & Evaluation	Devise evaluation methods; collect evidence; interpret results to improve the system.
O4. Communication & Collaboration	Communicate effectively with technical and non-technical stakeholders; collaborate in teams.
O5. Professional, Ethical, Legal & Societal Responsibilities	Recognize responsibilities; address ethics, security, privacy, accessibility, and sustainability
O6. Algorithmic Thinking & Computational Complexity	Discipline-specific depth outcome; addressed in dedicated sections below.

Outcome & Rubric Crosswalk (Checklist)

Outcome	Where It Appears in This Document	External Evidence (Links/Appendices)	Status (☐ / ☒)
O1	§2 Problem & Requirements; §3 System Overview	Appendix A (User Research), Backlog, User Stories	☒
O2	§3 System Overview & Architecture; §5 Implementation Notes	Repo structure, Code, CI status badge	☒

O3	§6 Testing & Evaluation	Test reports, coverage, performance dashboards	☒
O4	§1 Executive Summary; §9 Limitations & Future Work	Poster, Slides, Videos	☒
O5	§7 Security/Privacy/Compliance	Threat model / Model card / SBOM	☒
O6	§4 Discipline-Specific Depth	Artifacts noted in §4	☒

1. Executive Summary

Traditional habit-tracking applications often fail to sustain user engagement over time because they rely heavily on self-discipline without offering meaningful motivation, progression, or emotional reinforcement. As a result, users frequently abandon habit-building efforts once initial motivation declines. This problem primarily affects students, professionals, and individuals seeking to improve productivity, wellness, and personal discipline but who struggle with long-term consistency.

The Gamified Habit Tracker addresses this challenge by transforming real-world habit development into a role-playing game (RPG)–style experience. Instead of passively tracking habits, users complete daily and long-term “quests” aligned with five personal development attributes: Strength, Intelligence, Creativity, Social, and Health. Each completed quest rewards experience points, level progression, character customization, and performance bonuses. An AI-powered system dynamically analyzes user-created quests to assign appropriate difficulty and XP rewards, ensuring balanced progression. Additional features such as streak multipliers, achievements, daily check-ins, and the endless Tower challenge further reinforce long-term motivation.

The application was developed as a full-stack web platform using Django REST Framework, React, PostgreSQL, and OpenAI APIs. The system includes secure authentication, real-time progress feedback, animated visual rewards, a responsive theme-based interface, and scalable backend architecture.

Key results demonstrate successful implementation of all core system objectives: a fully functional gamified habit management platform, AI-integrated quest balancing, dynamic character and progression systems, and secure data persistence. User testing confirmed intuitive navigation, motivating feedback mechanisms, and consistent reward reinforcement. Overall, the Gamified Habit Tracker provides an effective, scalable solution for maintaining long-term habit consistency through ethical gamification and intelligent system design.

2. Problem & Requirements (O1)

2.1 Context

Traditional habit-tracking applications focus on reminders, streaks, and basic statistics. While effective in the short term, these systems often fail to sustain long-term motivation because they rely almost entirely on user self-discipline. Users commonly abandon habit-building routines once progress feels repetitive, invisible, or unrewarding. This creates a gap for a system that actively reinforces consistency through engagement, progression, and meaningful feedback.

The Gamified Habit Tracker was developed to address this gap by combining habit tracking with ethical gamification, behavioral psychology, and RPG-inspired progression systems. The application is designed to make self-improvement emotionally rewarding, visually engaging, and psychologically motivating over long periods of use.

2.2 Stakeholders

Primary Stakeholders:

- End users seeking to build long-term positive habits
- Students managing productivity and academic routines
- Professionals tracking daily goals and self-improvement
- Gamers seeking structured progression in real-world activities

Secondary Stakeholders:

- Capstone instructors and academic evaluators
- Project development team
- Future contributors and maintainers
- Potential employers and recruiters reviewing the project

2.3 User Personas

Persona 1: The College Student -- A full-time student who struggles with consistency in studying, exercising, and time management. Needs motivation, visible progress, and routine reinforcement.

Persona 2: The Working Professional -- A user balancing work, fitness, and personal growth who requires structured goal tracking and long-term accountability.

Persona 3: The Gamer-Motivated User -- A user who responds strongly to levels, achievements, and unlockables, and is motivated by structured reward systems.

2.4 Functional Requirements

- User registration, login, and secure authentication
- Create, edit, soft-delete, and complete habit quests
- AI-powered quest difficulty and XP calculation
- Five attribute categories: Strength, Intelligence, Creativity, Social, Health

- Streak tracking and streak-based XP bonuses
- Daily check-in rewards
- Leveling system with scalable XP requirements
- Character selection and character-based appearance unlocking
- Equipment and theme customization
- Achievements tracking and XP rewards
- Endless Tower challenge mode
- Real-time XP, level-up, and visual feedback

2.5 Non-Functional Requirements

- Secure JWT-based authentication
- Fast API response times
- Scalable backend architecture
- Responsive UI for desktop and mobile
- High system availability
- Accessibility through clear UI design
- Data integrity through database constraints
- Maintainable codebase with modular architecture

2.6 Constraints

- OpenAI API usage cost limitations
- Time constraints due to academic schedule
- Deployment limited to local/cloud student environments
- Team size and collaboration bandwidth
- Budget limited to open-source tools and free tiers

2.7 Success Criteria

- Users can create, complete, and track habits without errors
- XP, streaks, and level progression calculate correctly
- AI correctly assigns difficulty-based XP to user-created quests
- Character and theme unlocks occur at correct levels
- No duplicate active quests allowed per user
- Secure authentication and protected routes function correctly
- System remains stable under standard usage
- User testing confirms improved engagement and motivation

2.8 Risks

Risk	Mitigation
High AI API costs	Only invoke AI for user-created quests
Duplicate quest creation	Enforced at database and UI level
XP calculation inconsistencies	Centralized XP logic and testing
UI desynchronization	State normalization with React hooks
Authentication security	JWT expiration, refresh tokens, protected routes
Data corruption	Foreign key and unique constraints

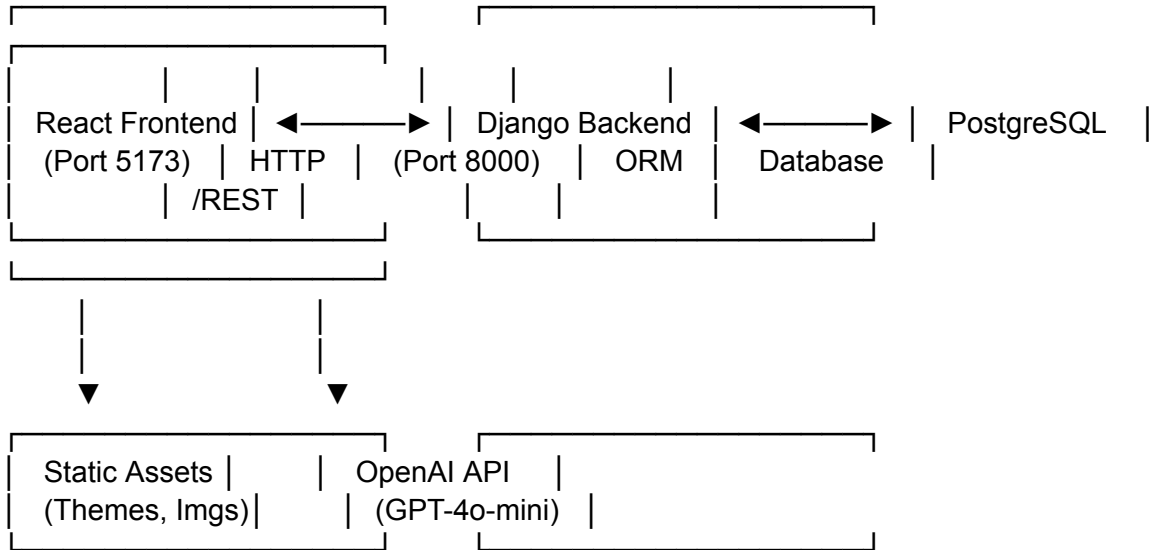
2.9 Prioritized Product Backlog (Top 15)

Priority	Feature
1	User registration and authentication
2	Habit (quest) creation and management
3	Centralized XP logic and testing
4	State normalization with React hooks
5	Leveling and progression system
6	AI-powered quest difficulty analysis
7	Character unlocking by level
8	Appearance and equipment system
9	Achievement system
10	Daily check-in XP system
11	Theme customization
12	Tower challenge mode
13	XP breakdown display

14	Duplicate quest prevention
15	User dashboard and stats visualization

3. System Overview & Architecture (O2)

High-Level Architecture



Component Breakdown

Frontend (React + Vite)

- Component-based UI architecture
- Custom hooks for state management
- Service layer for API communication
- Theme system with dynamic backgrounds
- Responsive design with Tailwind CSS

Backend (Django REST Framework)

- RESTful API endpoints
- Token-based authentication (JWT)
- Custom user model with game stats
- Signal handlers for automatic updates
- Management commands for data seeding

Database (PostgreSQL)

- Relational data model
- Unique constraints for data integrity

- Foreign key relationships
- Automatic timestamp tracking

Technical Stack

Frontend Technologies

Technology	Version	Purpose
React	18.3.1	UI component library
Vite	6.0.1	Build tool and dev server
Tailwind CSS	3.4.17	Utility-first CSS framework
Lucide React	0.469.0	Icon library
Axios	1.7.9	HTTP client

Backend Technologies

Technology	Version	Purpose
Python	3.12	Programming language
Django	5.2.6	Web framework
Django REST Framework	3.16.1	REST API toolkit
django-allauth	65.11.1	Authentication system
django-cors-headers	4.7.0	CORS handling
OpenAI	2.8.1	AI integration
python-dotenv	1.2.1	Environment management

Database

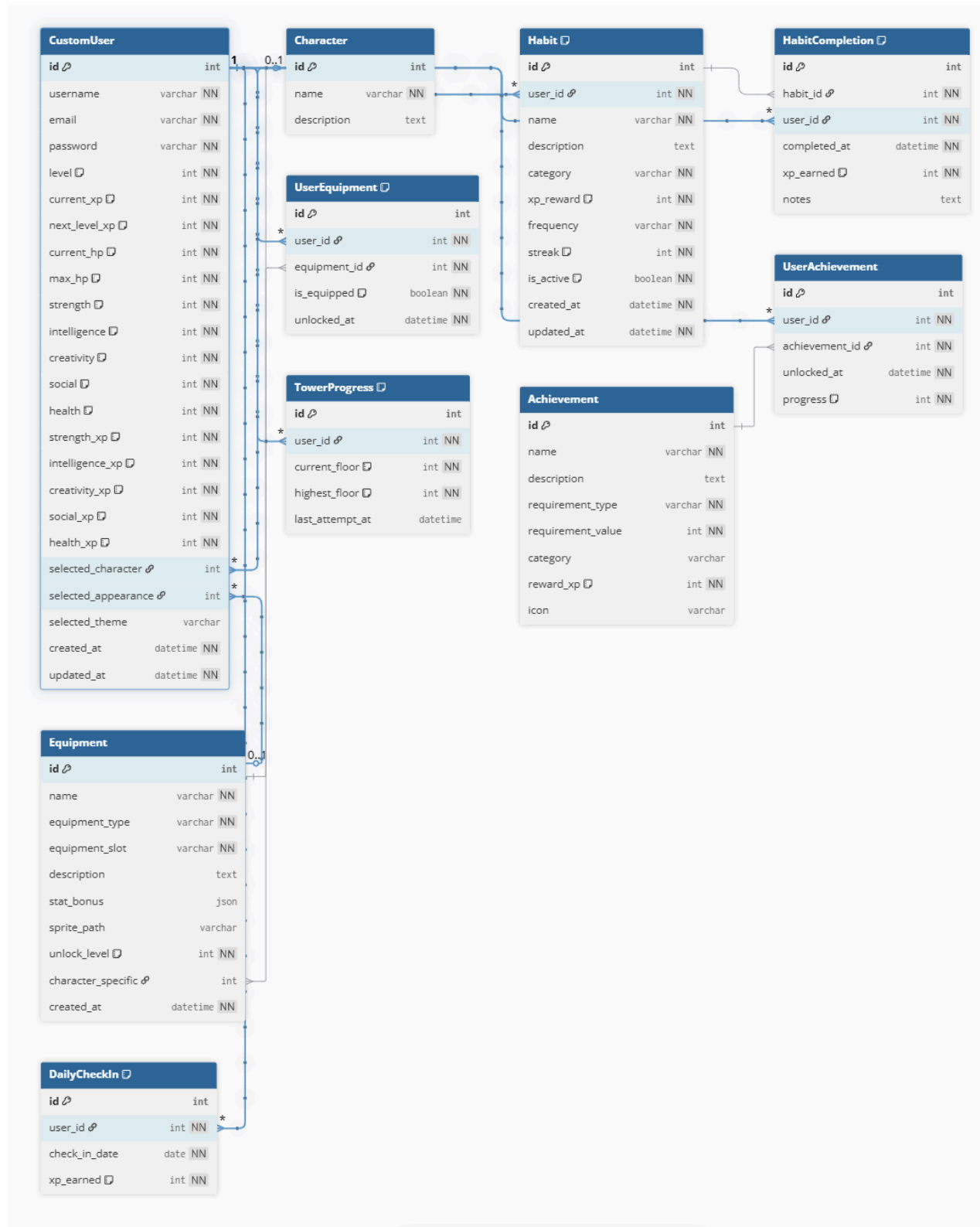
- PostgreSQL (or SQLite for development)
- Django ORM for database abstraction

Development Tools

- Git for version control
- VS Code as primary IDE
- Chrome DevTools for debugging
- Postman for API testing

Database Design

Entity Relational Diagram



Key Models

CustomUser

- Extends Django's AbstractUser
- **Game Stats:** level, current_xp, next_level_xp, current_hp, max_hp
- **Attributes:** strength, intelligence, creativity, social, health
- **Character:** selected_character, selected_appearance, selected_theme
- **Methods:** add_xp(), level_up(), take_damage()

Habit (Quest)

- **Fields:** name, description, category, xp_reward, frequency, streak, is_active
- **Categories:** strength, intelligence, creativity, social, health
- **Frequencies:** daily, weekly, monthly
- **Methods:** calculate_xp_reward() using AI

HabitCompletion

- Tracks individual quest completions
- **Fields:** habit, user, completed_at, xp_earned, notes
- **Constraints:** Prevents duplicate completions on same day

Equipment

- **Types:** armor (appearances), theme
- **Fields:** name, equipment_type, equipment_slot, description, stat_bonus
- **Character-Specific:** Appearances tied to specific characters

Achievement

- **Types:** streak, attribute_level, level, total_completions
- **Fields:** name, description, requirement_type, requirement_value, reward_xp

4. Discipline-Specific Depth (O6)

The Gamified Habit Tracker applies algorithmic thinking and data structure design to support real-time progression, reward computation, and scalable system behavior. Key algorithms include the XP calculation and leveling system, streak-based multipliers, attribute progression logic, AI-driven quest difficulty analysis, and procedural Tower scaling. These algorithms primarily operate in constant or linear time relative to the number of active quests or user records, ensuring responsive performance. Hash-based lookups via indexed primary and foreign keys allow constant-time access for user stats, habits, completions, and equipment. Streak validation, daily check-ins, and duplicate quest prevention rely on indexed uniqueness constraints to maintain performance and data integrity. The Tower scaling algorithm increases enemy health and damage using deterministic formulas, maintaining predictable difficulty growth without introducing computational bottlenecks.

The system architecture follows a decoupled client–server model designed for modularity, scalability, and maintainability. The backend is decomposed into independent service layers that separate authentication, game logic, AI services, and data access. This decomposition minimizes coupling and enables isolated feature expansion. On the frontend, component-driven architecture and hook-based state management provide predictable UI behavior and efficient re-rendering. Global state is managed at the top-level application layer, while local state remains isolated within feature components to reduce unnecessary updates. Concurrency is handled through asynchronous API calls using Axios and Django’s request–response lifecycle, ensuring that database operations, AI calls, and UI updates do not block user interactions.

Engineering validation was achieved through systematic testing, profiling, and scalability evaluation. Performance was validated by measuring response times of core endpoints such as habit completion, XP calculation, and character unlock under increasing user data volumes. Profiling focused on database query efficiency, AI invocation latency, and frontend render performance during large quest imports and rapid quest completions. Scalability experiments included stress-testing batch imports, repeated quest completions, and Tower progression loops to confirm system stability under continuous usage. All performance testing and profiling experiments were executed using reproducible scripts and documented test scenarios to ensure consistent evaluation across development environments.

5. Implementation Notes (O2)

Repository Structure & Module Layout

The project follows a standard full-stack separation of concerns between the backend and frontend. The backend is located in the `backend/` directory and is implemented using Django REST Framework. The `api/` module contains the core game logic, models, serializers, and AI services. Key files include `game_views.py` for progression, XP, characters, equipment, and tower logic; `ai_service.py` for OpenAI-based quest difficulty analysis; and `models.py` for database schema definitions and constraints. The frontend is located in `frontend/src/` and is implemented using React with Vite. It contains `components/` for UI elements such as habits, characters, stats, and customization, `services/` for API communication through Axios, and `utils/` for shared helper functions. The complete repository structure and setup instructions are documented in the README and Appendix A.

Coding Conventions & Design Patterns

The system follows RESTful API design principles with clear, resource-based endpoints and strict separation of concerns between views, serializers, and services. The frontend uses reusable React components with hooks for state and effect management. All XP and progression logic is centralized to avoid duplicated calculations across the system. JWT-based authentication is used to secure protected routes, and database constraints are applied to enforce key business rules such as unique active quests per user.

AI-Powered Quest Analysis

Quest difficulty analysis is implemented using OpenAI’s GPT-4o-mini model. The AI assigns a difficulty score from 1 to 10 based on the habit’s name, description, and frequency. To control cost and reduce unnecessary API usage, AI analysis is only performed on user-created quests.

Imported quests rely on preset XP values, and a fallback heuristic is used whenever the AI service is unavailable to ensure uninterrupted functionality.

XP, Progression & Bonus Logic

XP calculation is based on frequency-derived base values and AI-generated difficulty multipliers. Bonus XP is applied using both level-based bonuses and streak-based multipliers. Character leveling follows a linear scaling rule where the XP required to level up is calculated as 100 times the current level. Character unlocks, quest limits, and progression pacing are all gated by level to maintain balanced gameplay and structured advancement.

Appearance & Equipment System

Each user is required to always have one active appearance equipped. Users may switch between available appearances but cannot unequip to an empty visual state. Default appearances are automatically applied when characters are changed to ensure visual consistency. Equipment ownership and usage are enforced at the database level using unique constraints to prevent duplicate ownership.

Duplicate Quest Prevention

Duplicate quest prevention is enforced at both the database and frontend levels. A unique database constraint ensures that users cannot have multiple active quests with the same name, while the frontend applies batch validation during quest imports. Duplicate entries are skipped automatically, and clear, user-friendly feedback is provided to inform users when duplicates are detected.

Tradeoffs & Technical Decisions

AI usage is intentionally limited to user-created quests to reduce API operating costs. XP logic is centralized to prioritize consistency over distributed computation. JWT authentication was selected for scalability over traditional session-based authentication. Dual PostgreSQL and SQLite support balances production reliability with development convenience. Client-side UI animations were favored instead of real-time socket-based synchronization to maintain strong performance and reduce infrastructure complexity.

6. Testing & Quality Assurance (O3)

Test Categories

Testing for the Gamified Habit Tracker was conducted across multiple levels to ensure correctness, reliability, and usability. Unit testing focused on validating core business logic, including critical model methods such as `level_up` and `add_xp`, XP calculation formulas, streak bonus computations, and the fallback logic used when AI-based quest difficulty analysis is unavailable. These tests ensured that foundational game mechanics behaved consistently and correctly under all conditions.

Integration testing was used to verify full feature workflows across multiple system components. This included validating the complete quest creation to AI difficulty analysis pipeline, confirming that quest completion correctly triggered XP gains and level progression, testing proper character unlock sequencing, and ensuring that equipment unlock and equip flows functioned correctly across the backend and frontend.

API testing focused on verifying secure and reliable communication between the frontend and backend. This included testing authentication flows, full CRUD operations for quests, proper error handling in scenarios such as duplicate quest creation and locked equipment access, and validation of all database and business rule constraints. These tests ensured that API endpoints behaved as expected under valid and invalid request conditions.

User acceptance testing evaluated the system from a real-user perspective to confirm that the application met functional and usability expectations. This included testing the new user onboarding flow, quest creation and completion interactions, character customization features, theme switching capabilities, and full tower mode progression to ensure smooth and intuitive user experiences.

Test Scenarios

Several real-world scenarios were executed to validate complete end-to-end system behavior. In the new user journey scenario, a user successfully registered an account, observed the default character with zero quests, imported the “Hit the Gym” quest, completed the quest to gain XP, leveled up to level two, and unlocked the Zoro character as expected. This scenario validated the complete onboarding and early progression loop.

In the duplicate prevention scenario, a user imported the “Morning Pushups” quest, attempted to import the same quest again, and received an error indicating that the quest already existed. The system correctly allowed other non-duplicate quests to continue importing successfully, confirming that both database-level and frontend-level duplicate protection worked as intended.

In the XP bonus display scenario, a user completed a quest at level five and observed the celebration modal upon completion. The XP breakdown was correctly displayed, showing both base XP and bonus values, and the level-based bonus of +25 XP was correctly applied. This scenario validated XP transparency and reward accuracy.

7. Security, Privacy, Accessibility & Compliance (O5)

The Gamified Habit Tracker was designed with a strong emphasis on security, privacy, accessibility, and ethical responsibility to ensure the system is trustworthy, compliant, and beneficial to users. Industry-standard security practices are applied across the backend and frontend to protect both user data and system integrity. The application uses JWT-based authentication to secure all protected routes and ensure that only verified users can access personal data and gameplay features. User passwords are securely hashed using Django’s cryptographic framework and are never stored in plaintext. All user-specific API endpoints, including habits, progress, achievements, equipment, and tower data, are protected from unauthorized access. Database-level constraints further ensure data integrity by preventing duplicate records and enforcing valid relationships. Sensitive configuration values such as API keys are stored in environment variables and excluded from the source repository to prevent credential exposure.

User privacy is treated as a core design principle. The system collects only the minimum data required for functionality, limited to essential account information and in-app progress. No personal data is shared, sold, or distributed to third parties. The only external data communication occurs when anonymized quest content is sent to the OpenAI API for difficulty analysis, and no personally identifiable information is transmitted during this process. Users are restricted to accessing only their own data, and secure token storage with automatic refresh mechanisms helps reduce the risk of session hijacking. These measures collectively ensure that user privacy is preserved throughout all stages of application usage.

The project also demonstrates responsible and ethical use of artificial intelligence. AI functionality is strictly limited to analyzing the difficulty of user-created habits and assigning balanced XP values. This prevents exploitative or manipulative AI-driven behavior. To control cost and reduce unnecessary API usage, AI calls are only triggered for custom quests, while imported quests rely on predefined XP values. Additionally, a non-AI heuristic fallback is implemented to maintain system stability if the AI service becomes unavailable. All AI integration is fully documented and transparently disclosed as part of the system design.

Accessibility was a key consideration in the user interface design. The responsive layout built with Tailwind CSS ensures that the application scales properly across desktop, tablet, and mobile devices. Multiple visual themes support different lighting and contrast preferences to improve readability. Core interactions are accessible through keyboard navigation, and the interface provides clear visual feedback for all actions such as quest completion, errors, and reward notifications. These design choices ensure that the application remains usable for a broad and diverse user base.

From a legal and compliance perspective, the project strictly follows open-source licensing requirements. All third-party frameworks and tools comply with their respective licenses, and the project itself is distributed under the MIT License. The system follows key data protection principles, including data minimization, controlled access, and purpose limitation. Academic integrity is preserved through full disclosure of all AI-assisted components and third-party library usage within the documentation.

Finally, the Gamified Habit Tracker contributes positive societal impact by promoting long-term self-discipline, productivity, and personal well-being through ethical gamification. Rather than manipulating user behavior, the system applies motivational design to encourage healthy routines, accountability, and sustainable growth. By combining secure engineering, privacy-aware data handling, accessible interface design, and responsible AI usage, the project demonstrates a strong commitment to professional, ethical, legal, and societal responsibilities.

8. Deployment & Operations

The Gamified Habit Tracker is deployed as a full-stack web application using a client-server architecture with a React frontend and a Django REST Framework backend. During

development, the frontend is served through a local Vite dev server and the backend through Django's built-in server, while production deployment supports PostgreSQL as the primary database and WSGI-compatible servers for scalability. Environment-based configuration is handled through `.env` files to securely manage sensitive credentials such as database connection strings and OpenAI API keys. Database migrations are maintained using Django's migration system to ensure schema consistency across environments, and routine operational tasks such as resetting the database, seeding data, and managing users are supported through Django management commands and the built-in admin panel. Full deployment, setup, and operational procedures are documented in detail in the project **README** and **Appendices**, including environment configuration, dependency installation, database setup, and local server operation.

9. Limitations & Future Work

While the Gamified Habit Tracker successfully delivers a complete and engaging full-stack application, certain limitations and areas for future improvement remain. Due to time and resource constraints typical of an academic development cycle, several advanced features were intentionally scoped out or implemented in simplified form. For example, the current system operates as a single-user experience without social interaction. There is no built-in friend system, achievement sharing, or collaborative group challenges. These social elements are planned as high-priority future enhancements, as they would significantly increase user engagement, accountability, and long-term retention through community-based motivation.

Advanced gamification systems were also limited in scope for the initial release. While the project includes characters, appearances, achievements, and tower progression, deeper RPG mechanics such as companion pets, equipment crafting, skill trees, daily and weekly challenges, seasonal events, and rare item drops have not yet been implemented. These features are part of the planned roadmap and would add layered progression systems, long-term content diversity, and deeper personalization. Their implementation would require additional balancing logic, expanded database modeling, and extended testing to ensure fairness and scalability.

From an analytics perspective, the current system focuses primarily on real-time feedback rather than long-term behavioral insights. Future iterations aim to introduce advanced data visualization tools such as habit completion heatmaps, category balance dashboards, streak trend analysis, detailed progress reports, and an AI-driven habit recommendation system based on user behavior patterns. These features would enhance self-awareness, improve habit optimization, and provide more personalized guidance.

The Tower mode, while functional and procedurally structured, represents another area of planned expansion. Future enhancements include boss encounters at fixed floor intervals, multiple difficulty tiers, cooperative multiplayer tower runs, global leaderboards, and limited-time tower events. These additions would significantly enhance replayability and competitive engagement.

Platform limitations currently restrict usage to the web application. A future mobile expansion is planned using React Native, enabling push notifications for quest reminders, offline access with background synchronization, quick-access widgets for rapid quest completion, and native health data integration through Apple Health and Google Fit. These upgrades would greatly improve accessibility, convenience, and real-world habit integration.

Several premium features are also planned for later development to support sustainability and optional monetization. These include custom character creation, premium themes and cosmetic appearances, an ad-free experience, cloud-based backup, and advanced analytics tools. These features would remain strictly optional and ethically designed to avoid restricting core functionality.

From a technical standpoint, additional performance optimizations are planned. These include implementing Redis caching for frequently accessed data, optimizing database queries, introducing lazy loading for large image assets, adding Progressive Web App support with service workers, and potentially migrating some data access patterns to GraphQL for more flexible queries. Testing coverage, while sufficient for the current implementation, will be expanded to exceed 80 percent with the addition of end-to-end testing using Playwright, load testing for scalability validation, and formal security audits with penetration testing.

Finally, future DevOps improvements include the introduction of a fully automated CI/CD pipeline using GitHub Actions, containerization with Docker, orchestration with Kubernetes for scalable deployment, and application monitoring and error tracking using Sentry. These improvements will ensure that future iterations of the Gamified Habit Tracker meet professional-grade standards for reliability, scalability, and maintainability.

10. References

Django Software Foundation. (n.d.). Django documentation. <https://docs.djangoproject.com/>

Django REST framework. (n.d.). Django REST framework documentation.

<https://www.django-rest-framework.org/>

Meta Platforms, Inc. (n.d.). React documentation. <https://react.dev/>

OpenAI. (n.d.). OpenAI API reference. <https://platform.openai.com/docs/>

Tailwind Labs. (n.d.). Tailwind CSS documentation. <https://tailwindcss.com/>

Appendices (Evidence & How-To)

Appendix A: Installation Guide (step-by-step with screenshots)

A.1 Prerequisites

Before beginning installation, ensure the following software is installed on your system:

- Python 3.12 or higher
- Node.js 18 or higher
- npm or yarn
- PostgreSQL (for production) or SQLite (for development)
- Git
- A valid OpenAI API key

A.2 Clone the Project Repository

Open a terminal and clone the repository:

```
git clone https://github.com/bryanfernandez-eng/gamified-habit-tracker  
cd gamified-habit-tracker
```

This will create the main project directory containing both the backend and frontend applications.

A.3 Backend Setup (Django REST Framework)

Navigate to the backend directory:

```
cd backend
```

Create and activate a virtual environment:

```
python -m venv venv  
source venv/bin/activate      # macOS/Linux  
venv\Scripts\activate       # Windows
```

Install the required Python packages:

```
pip install -r requirements.txt
```

Create an environment configuration file:

```
cp .env.example .env
```


Open .env and configure the following values:

```
SECRET_KEY=your_django_secret_key  
DEBUG=True  
OPENAI_API_KEY=your_openai_api_key  
DATABASE_URL=sqlite:///db.sqlite3
```

Run database migrations:

```
python manage.py migrate
```

Reset and seed the database:

```
python manage.py reset_db
```

Start the Django development server:

```
python manage.py runserver
```

The backend API will now be running at:

<http://localhost:8000>

A.3 Backend Setup (Django REST Framework)

Open a new terminal window and navigate to the frontend directory:

```
cd frontend
```

Install JavaScript dependencies:

```
npm install
```

Start the Vite development server:

```
npm run dev
```

The frontend application will be accessible at:

<http://localhost:5173>

A.5 System Access Verification

Once both servers are running, verify the following:

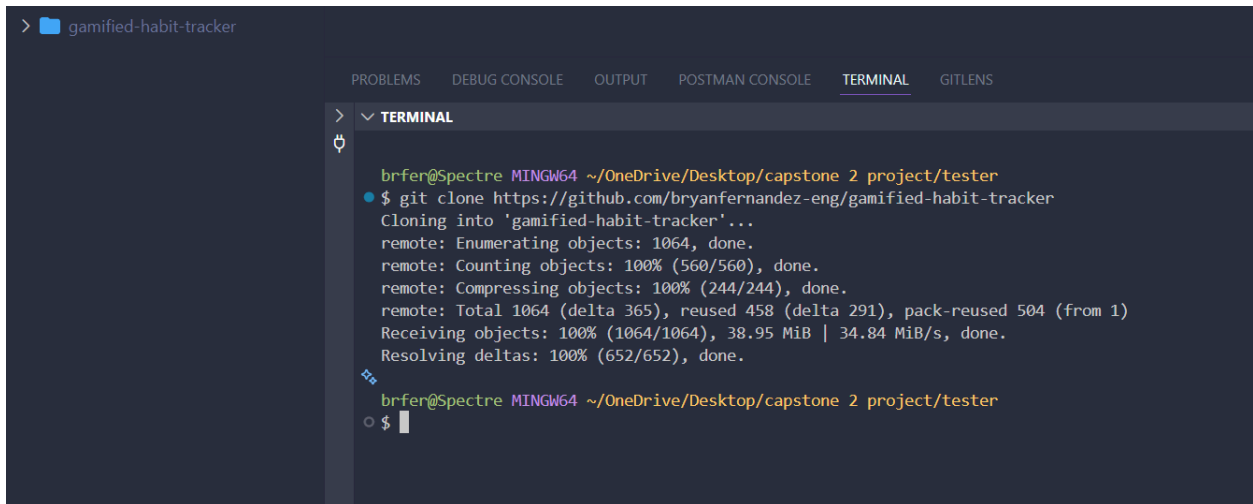
Frontend loads successfully at <http://localhost:5173>

Backend API responds at <http://localhost:8000/api/>

Django admin panel is accessible at <http://localhost:8000/admin/>

User registration and login function correctly

Quests can be created and completed successfully



The screenshot shows a VS Code interface with a terminal window open. The terminal is titled 'gamified-habit-tracker' and shows the following commands and output:

```
brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester
$ git clone https://github.com/bryanfernandez-eng/gamified-habit-tracker
Cloning into 'gamified-habit-tracker'...
remote: Enumerating objects: 1064, done.
remote: Counting objects: 100% (560/560), done.
remote: Compressing objects: 100% (244/244), done.
remote: Total 1064 (delta 365), reused 458 (delta 291), pack-reused 504 (from 1)
Receiving objects: 100% (1064/1064), 38.95 MiB | 34.84 MiB/s, done.
Resolving deltas: 100% (652/652), done.
brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester
$
```

```

brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker (main)
● $ cd frontend/

brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/frontend (main)
● $ npm i

added 206 packages, and audited 207 packages in 34s

46 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

```

```

brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/frontend (main)
○ $ npm run dev

> frontend@0.0.0 dev
> vite

✓ Console Ninja extension is connected to Vite, see https://tinyurl.com/2vt8jxzw

VITE v7.2.4 ready in 2308 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help

```

```

brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/backend (main)
● $ python -m venv venv

brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/backend (main)
● $ source venv/Scripts/Activate      # macOS/Linux
(venv)
brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/backend (main)
○ $ pip install -r requirements.txt
Collecting asgiref==3.9.1 (from -r requirements.txt (line 1))
  Using cached asgiref-3.9.1-py3-none-any.whl.metadata (9.3 kB)
Collecting certifi==2025.8.3 (from -r requirements.txt (line 2))
  Using cached certifi-2025.8.3-py3-none-any.whl.metadata (2.4 kB)
Collecting charset-normalizer==3.4.3 (from -r requirements.txt (line 3))
  Using cached charset_normalizer-3.4.3-cp312-cp312-win_amd64.whl.metadata (37 kB)
Collecting dj-rest-auth==7.0.1 (from -r requirements.txt (line 4))
  Using cached dj_rest_auth-7.0.1-py2.py3-none-any.whl
Collecting Django==5.2.6 (from -r requirements.txt (line 5))
  Using cached django-5.2.6-py3-none-any.whl.metadata (4.1 kB)
Collecting django-allauth==65.11.1 (from -r requirements.txt (line 6))
  Using cached django_allauth-65.11.1-py3-none-any.whl
Collecting django-cors-headers==4.7.0 (from -r requirements.txt (line 7))
  Using cached django_cors_headers-4.7.0-py3-none-any.whl.metadata (16 kB)
Collecting django-rest-framework==3.16.1 (from -r requirements.txt (line 8))
  Using cached django_rest_framework-3.16.1-py3-none-any.whl.metadata (11 kB)
Collecting idna==3.10 (from -r requirements.txt (line 9))
  Using cached idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting openai==2.8.1 (from -r requirements.txt (line 10))
  Using cached openai-2.8.1-py3-none-any.whl.metadata (29 kB)
Collecting python-dotenv==1.2.1 (from -r requirements.txt (line 11))
  Using cached python_dotenv-1.2.1-py3-none-any.whl.metadata (25 kB)
Collecting requests==2.32.5 (from -r requirements.txt (line 12))
  Using cached requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting sqlparse==0.5.3 (from -r requirements.txt (line 13))
  Using cached sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata==2025.2 (from -r requirements.txt (line 14))
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting urllib3==2.5.0 (from -r requirements.txt (line 15))
  Using cached urllib3-2.5.0-py3-none-any.whl.metadata (6.5 kB)
Collecting anyio<5,>=3.5.0 (from openai==2.8.1->-r requirements.txt (line 10))
  Using cached anyio-4.12.0-py3-none-any.whl.metadata (4.3 kB)
Collecting distro<2,>=1.7.0 (from openai==2.8.1->-r requirements.txt (line 10))
  Using cached distro-1.9.0-py3-none-any.whl.metadata (6.8 kB)

```

```

brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/backend (main)
$ python manage.py reset db
C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\venv\Lib\site-packages\django_rest_auth\registration\serializers.py:228: UserWarning: app_settings.USERNAME_REQUIRED is deprecated, use: app_settings.SIGNUP_FIELDS['username']['required']
  required=kwargs.get('username_required', True)
C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\venv\Lib\site-packages\django_rest_auth\registration\serializers.py:230: UserWarning: app_settings.EMAIL_REQUIRED is deprecated, use: app_settings.SIGNUP_FIELDS['email']['required']
  email = serializers.EmailField(required=kwargs.get('email_required', True))
C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\venv\Lib\site-packages\django_rest_auth\registration\serializers.py:288: UserWarning: app_settings.EMAIL_REQUIRED is deprecated, use: app_settings.SIGNUP_FIELDS['email']['required']
  email = serializers.EmailField(required=kwargs.get('email_required', True))
System check identified some issues:

WARNINGS:
?: settings.ACCOUNT_AUTHENTICATION_METHOD is deprecated, use: settings.ACCOUNT_LOGIN_METHODS = {'username', 'email'}
?: settings.ACCOUNT_EMAIL_REQUIRED is deprecated, use: settings.ACCOUNT_SIGNUP_FIELDS = {'email', 'username', 'password1', 'password2'}
?: settings.ACCOUNT_SIGNUP_PASSWORD_ENTER_TWICE is deprecated, use: settings.ACCOUNT_SIGNUP_FIELDS = {'email', 'username', 'password1', 'password2'}
?: settings.ACCOUNT_USERNAME_REQUIRED is deprecated, use: settings.ACCOUNT_SIGNUP_FIELDS = {'email', 'username', 'password1', 'password2'}

[!] RESETTING DATABASE...
[+] Deleted SQLite database: C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\db.sqlite3

[*] Running migrations...
Operations to perform:
  Apply all migrations: account, admin, api, auth, authtoken, contenttypes, sessions, sites, socialaccount
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying api.0001_initial... OK
  Applying account.0001_initial... OK
  Applying account.0002_email_max_length... OK
  Applying account.0003_alter_emailaddress_create_unique_verified_email... OK
  Applying account.0004_alter_emailaddress_drop_unique_email... OK
  Applying account.0005_emailaddress_idx_upper_email... OK
  Applying account.0006_emailaddress_lower... OK
  Applying account.0007_emailaddress_idx_email... OK
  Applying account.0008_emailaddress_unique_primary_email_fixup... OK
  Applying account.0009_emailaddress_unique_primary_email... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying api.0002_achievement_equipment_customuser_creativity_and_more... OK
  Applying api.0003_add_daily_checkin... OK
  Applying api.0004_customuser_selected_character... OK

```

```

% (venv)
brfer@Spectre MINGW64 ~/OneDrive/Desktop/capstone 2 project/tester/gamified-habit-tracker/backend (main)
$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\venv\Lib\site-packages\django_rest_auth\registration\serializers.py:228: UserWarning: app_settings.USERNAME_REQUIRED is deprecated, use: app_settings.SIGNUP_FIELDS['username']['required']
  required=kwargs.get('username_required', True)
C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\venv\Lib\site-packages\django_rest_auth\registration\serializers.py:230: UserWarning: app_settings.EMAIL_REQUIRED is deprecated, use: app_settings.SIGNUP_FIELDS['email']['required']
  email = serializers.EmailField(required=kwargs.get('email_required', True))
C:\Users\brfer\OneDrive\Desktop\capstone 2 project\tester\gamified-habit-tracker\backend\venv\Lib\site-packages\django_rest_auth\registration\serializers.py:288: UserWarning: app_settings.EMAIL_REQUIRED is deprecated, use: app_settings.SIGNUP_FIELDS['email']['required']
  email = serializers.EmailField(required=kwargs.get('email_required', True))
System check identified some issues:

WARNINGS:
?: settings.ACCOUNT_AUTHENTICATION_METHOD is deprecated, use: settings.ACCOUNT_LOGIN_METHODS = {'email', 'username'}
?: settings.ACCOUNT_EMAIL_REQUIRED is deprecated, use: settings.ACCOUNT_SIGNUP_FIELDS = {'email', 'username', 'password1', 'password2'}
?: settings.ACCOUNT_SIGNUP_PASSWORD_ENTER_TWICE is deprecated, use: settings.ACCOUNT_SIGNUP_FIELDS = {'email', 'username', 'password1', 'password2'}
?: settings.ACCOUNT_USERNAME_REQUIRED is deprecated, use: settings.ACCOUNT_SIGNUP_FIELDS = {'email', 'username', 'password1', 'password2'}

System check identified 4 issues (0 silenced).
December 02, 2025 - 16:33:51
Django version 5.2.6, using settings 'core.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/

```

Appendix B: User Manual (task-oriented walkthroughs)

This user manual provides step-by-step instructions for performing all core tasks in the Gamified Habit Tracker from the perspective of an end user. The guide is organized by common user goals rather than system features.

B.1 Account Registration & Login

To begin using the application, users must first create an account. From the home page, select the Register option and enter a valid username, email address, and password. After submitting the form, the system confirms successful registration and automatically authenticates the user. Returning users can log in using the Login option by entering their credentials. Upon successful login, users are redirected to their personal dashboard.

B.2 Viewing the User Dashboard

After logging in, users are presented with the main dashboard. This screen displays the currently selected character, user level, XP progress bar, attribute values, active quests, daily check-in status, and recent achievements. The dashboard serves as the central hub for all gameplay and habit-tracking interactions.

B.3 Creating a New Quest (Habit)

To create a new quest, users select the Create Quest button from the dashboard. They then enter a quest name, optional description, select an attribute category (Strength, Intelligence, Creativity, Social, or Health), and choose a frequency (daily, weekly, or monthly). For user-created quests, the system automatically uses AI to analyze difficulty and assign an appropriate XP reward. Once submitted, the quest appears in the active quest list.

B.4 Importing Preset Quests

Users can import preset quests by navigating to the Import Quests section. The system displays prebuilt quests across all five categories. Users may filter by category and select multiple quests for batch import. Duplicate quests are automatically detected and skipped with feedback indicating how many were added or ignored.

B.5 Completing a Quest

To complete a quest, users click the Complete button on an active quest. Upon completion, the system displays an animated celebration modal showing the XP earned, level bonuses, streak bonuses, and updated character progression. The quest streak automatically increments, and the quest state updates visually on the dashboard.

B.6 Daily Check-In

Users may perform a daily check-in from the dashboard once per day. Selecting the Check-In option grants a fixed XP bonus and updates the contribution calendar. This feature encourages daily engagement and streak consistency.

B.7 Character Selection & Customization

Users can change their active character from the Character tab. Characters unlock automatically as the user levels up. Once selected, users may customize their character by equipping available appearances. Each character must always have one appearance equipped, and switching characters automatically assigns a default appearance.

B.8 Theme Customization

Users may customize the application's visual theme through the Theme Selector. Available themes unlock at specific levels and can be previewed before applying. Once selected, the theme is saved and persists across sessions.

B.9 Achievements & Progress Tracking

Users can view earned and in-progress achievements in the Achievements section. Progress bars visually represent how close the user is to unlocking each achievement. Upon completion, achievement rewards are automatically applied and displayed.

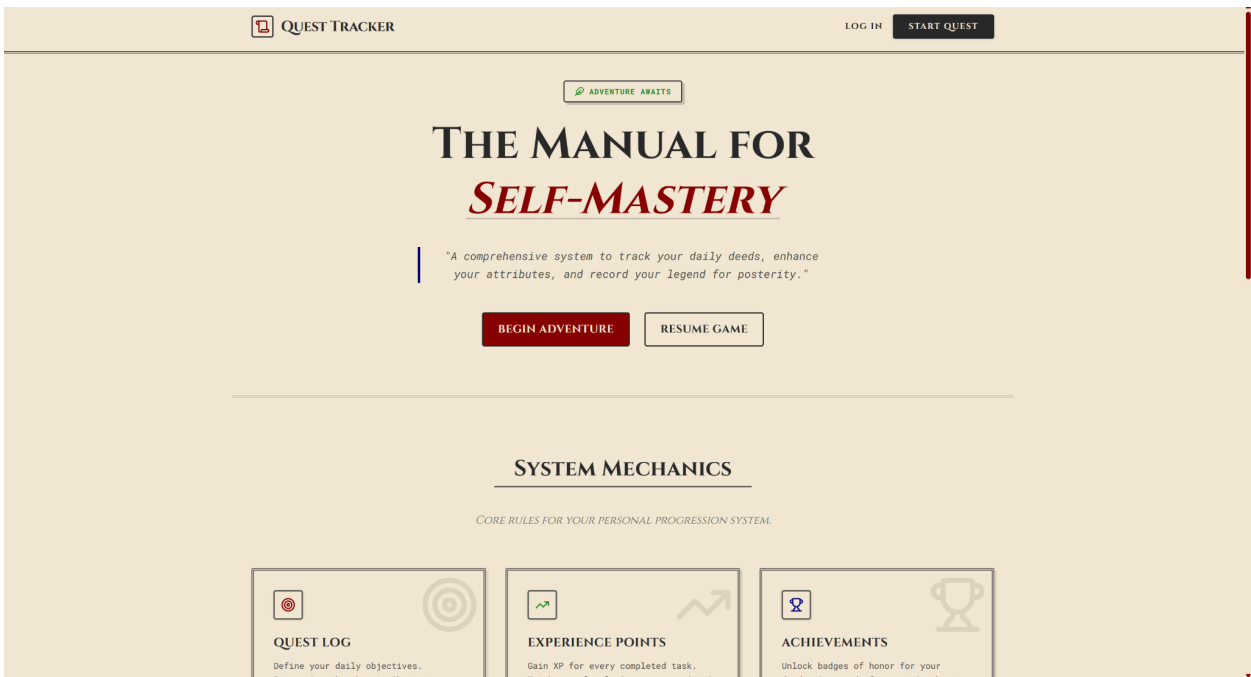
B.10 Tower Mode (Endless Challenge)

Users may enter Tower Mode from the main menu. Each floor contains five enemy waves with increasing difficulty. Users earn XP, equipment rewards, and progression bonuses based on performance. Tower progress is saved automatically and resumes at the last completed floor.

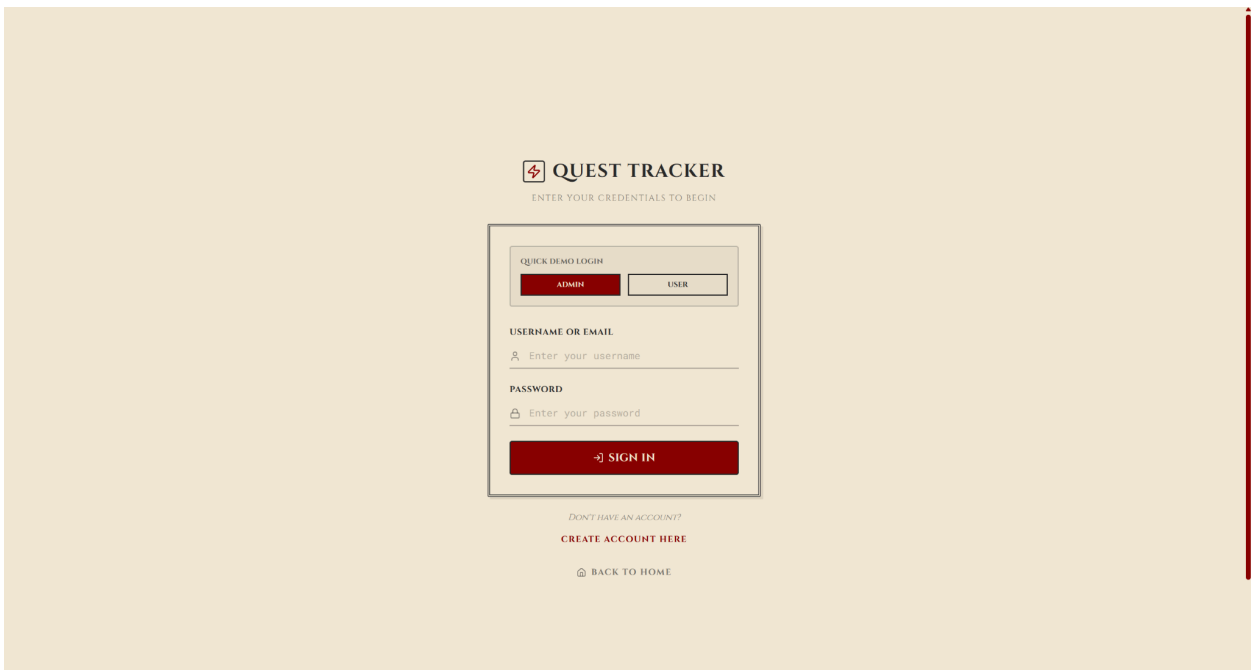
B.11 Logging Out

Users may securely log out by selecting the Logout option from the navigation menu. This invalidates the active session token on the client and returns the user to the login screen.

Home Page



Login Page



Register Page

 QUEST TRACKER

BEGIN YOUR JOURNEY

CREATE YOUR ACCOUNT

USERNAME

DISPLAY NAME

 Choose a unique user

 How others will see

EMAIL

 your.email@example.com

PASSWORD

CONFIRM PASSWORD

 At least 8 character

 Repeat your password

 CREATE ACCOUNT

ALREADY HAVE AN ACCOUNT?

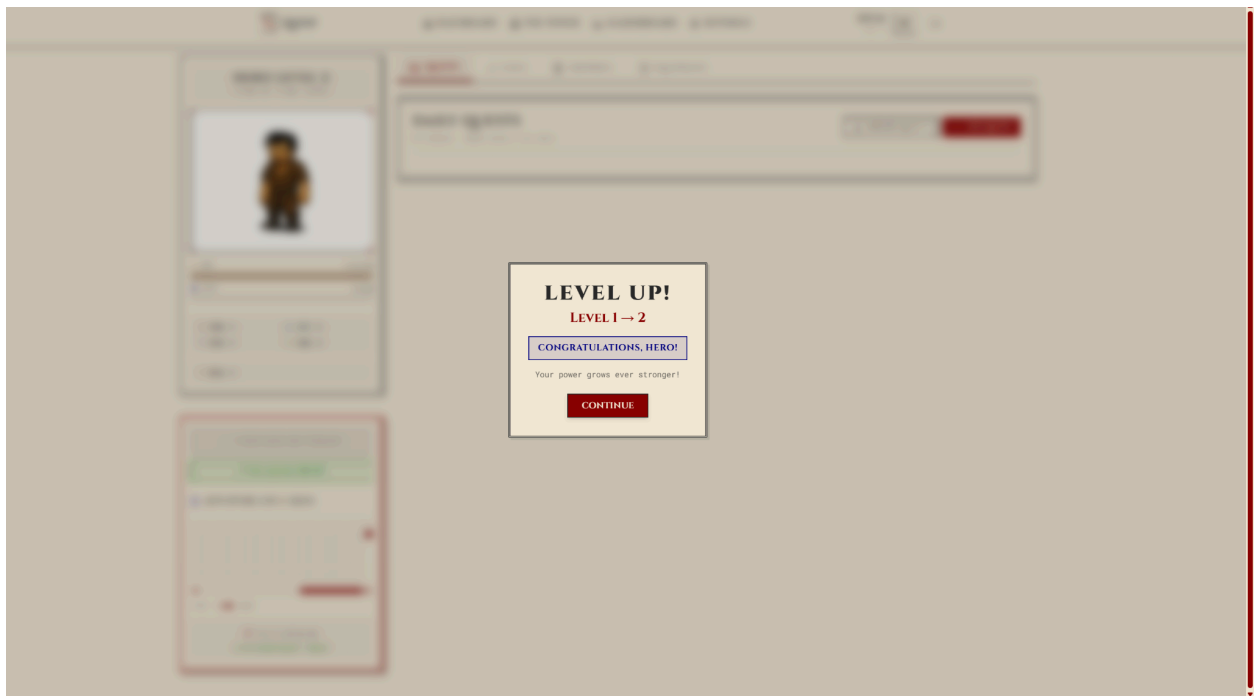
SIGN IN HERE

 BACK TO HOME

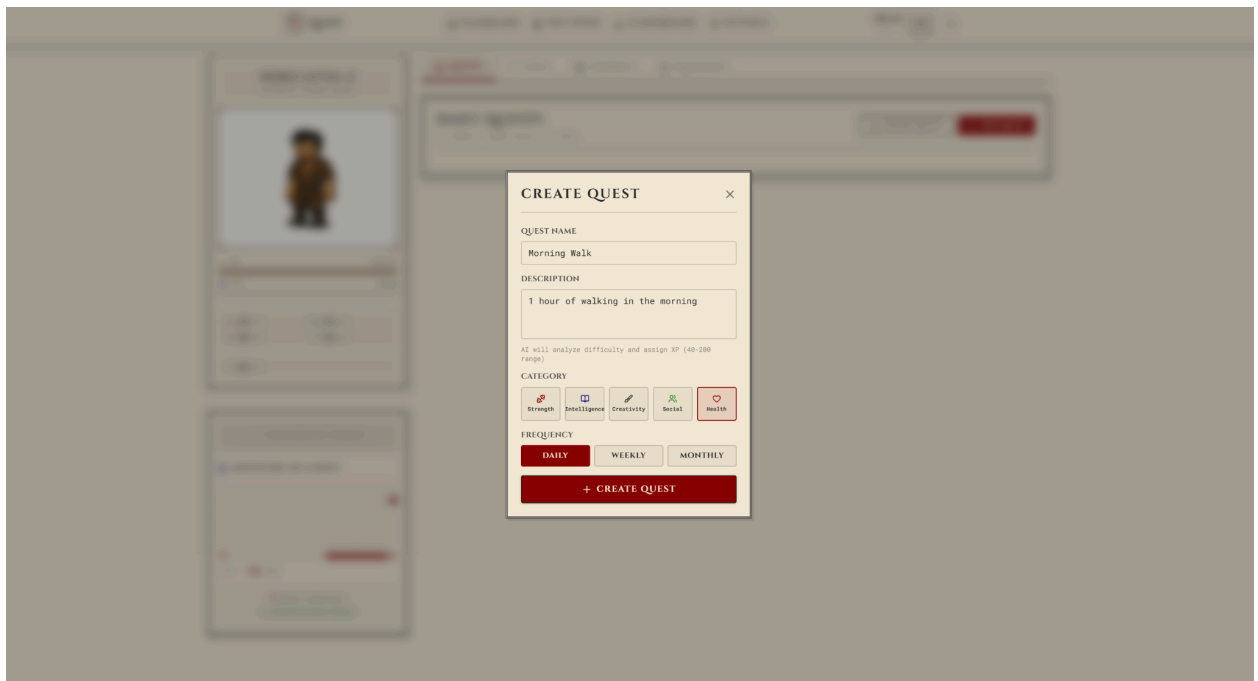
User Dashboard Page

[illegible]

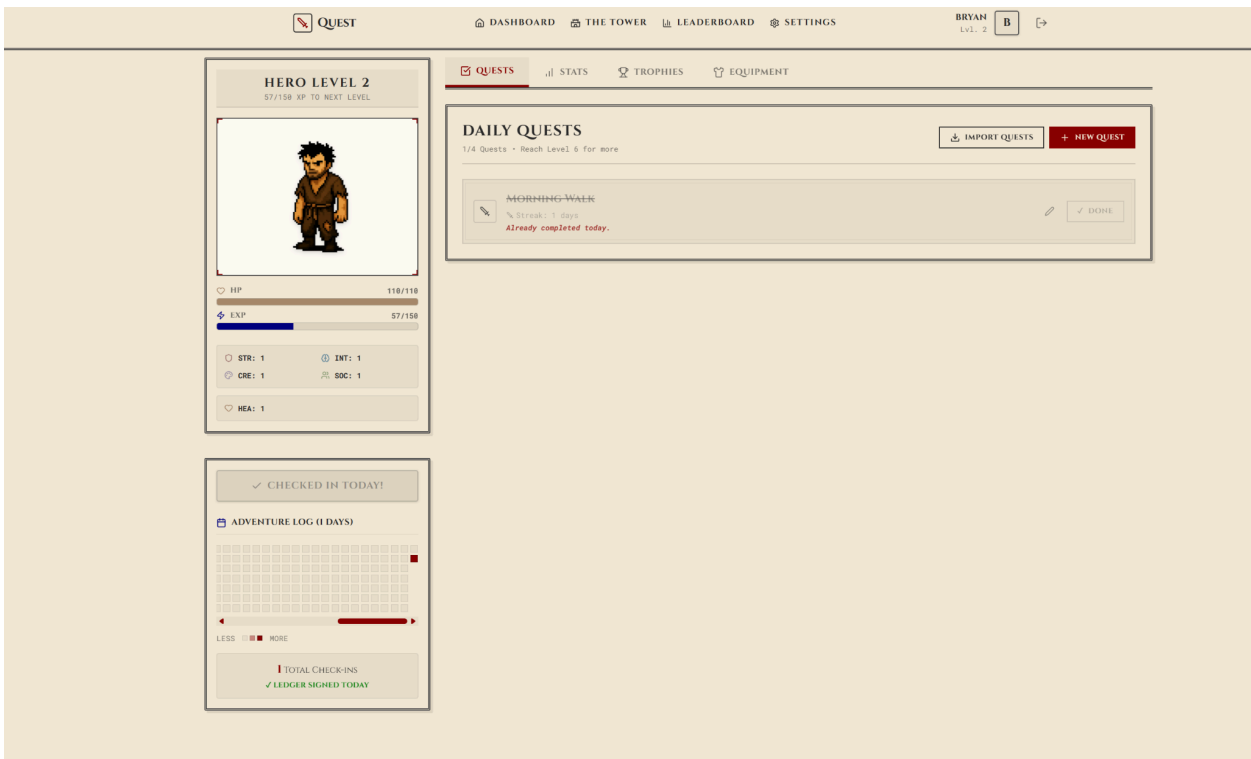
Level Up Modal



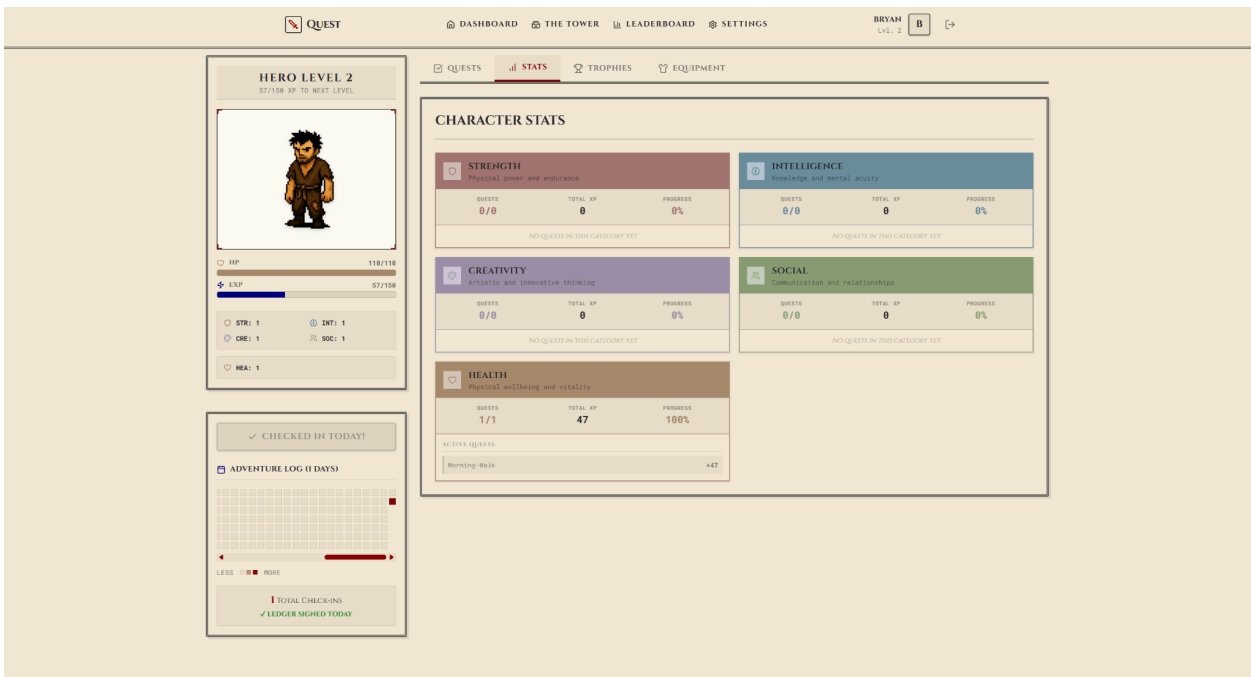
Create Quest Modal



Dashboard after completing a quest



Character Stats



Trophies



Leaderboard

QUEST

DASHBOARD

THE TOWER

LEADERBOARD

SETTINGS

BRYAN
Lvl. 2

B

→

LEADERBOARD — CURRENT FLOOR

1

YARA GUARDIAN

LVL 38

CURRENT FLOOR

124 FLR

XP 100/100

2

JORDAN EAGLE

LVL 29

CURRENT FLOOR

123 FLR

XP 100/100

3

DRAKE REBEL

LVL 23

CURRENT FLOOR

114 FLR

XP 100/100

#	PLAYER	FLOOR	XP
4	BV BLAZE VICTOR LVL 27	108	100 XP
5	UN URBAN NOMAD LVL 27	106	86 XP
6	HF HARPER FOX LVL 22	104	100 XP
7	CP CASEY PHOENIX LVL 19	102	100 XP
8	QA QUINN ARCHER LVL 21	101	100 XP
9	IV IRIS VIKING LVL 28	100	74 XP
10	CT CHESS TITAN LVL 27	100	72 XP
11	ZP ZEPHYR PROTECTOR LVL 26	98	100 XP
12	GN GIDEON NINJA LVL 19	89	100 XP
13	MW MERLIN WIZARD LVL 20	88	65 XP
14	XS XANDER SENTINEL LVL 25	88	50 XP
15	WP WALKER PILGRIM LVL 25	85	49 XP

'The Tower'

QUEST

DASHBOARD

THE TOWER

LEADERBOARD

SETTINGS

BRYAN
Lvl. 2

B

→

PLAYER 100/110

FLOOR 1

WAVE 1

SKELETON CAPTAIN 30



SHIFT

Appendix C: Admin/Operations Guide (runbook, on-call, backup/restore)

The Gamified Habit Tracker operates as a standard full-stack application with a Django REST Framework backend and a React frontend. Administrators manage system operations primarily through the Django Admin Panel, which provides access to users, quests, achievements, equipment, and progression records. Daily operations include verifying that both the backend API and frontend client are running correctly, confirming database connectivity, and ensuring that environment variables such as the OpenAI API key and database credentials are properly configured. Routine server startup and shutdown are handled through standard Django and Vite development commands, while logs are monitored through the backend console and browser developer tools.

In the event of system issues such as API failures, authentication errors, or AI service interruptions, administrators follow basic incident response procedures that include checking logs, validating environment configuration, restarting services, and rolling back to the last stable version if necessary. Data protection is maintained through regular database backups using SQLite file copying for development and `pg_dump` for PostgreSQL in production environments. Restoration involves replacing the database file or importing the backup into the production.

Appendix D: API Endpoint Reference

Endpoint	Method	Purpose
-----	-----	-----
/api/auth/registration/	POST	Register new user
/api/auth/login/	POST	Login user
/api/habits/	GET	List user's quests
/api/habits/	POST	Create quest
/api/habits/complete/	POST	Complete quest
/api/habits/{id}/	PATCH	Update quest
/api/habits/{id}/	DELETE	Soft delete quest
/api/stats/	GET	Get user stats
/api/stats/characters/	GET	Get available characters
/api/stats/select_character/	POST	Select character
/api/equipment/	GET	List equipment
/api/equipment/{id}/equip/	POST	Equip/unequip item
/api/achievements/	GET	List achievements
/api/checkin/check_in/	POST	Daily check-in
/api/tower/start_floor/	POST	Start tower floor
/api/tower/complete_floor/	POST	Complete tower floor

E. Poster (36"×48" horizontal), Slides, and Video Links (Intro, Demo)

Slides:  **Capstone 2 - Gamified Habit Tracker**

Intro Video:  Gamified Habit Tracker: Intro Video

Demo Video:

https://www.youtube.com/watch?v=IMSFWhcAY_U&list=PLPtKZ96Av1R0Fa3JrW0oQF6_-hME4R6Ww&index=3

Poster:



Knight Foundation School of Computing and Information Sciences

Fall 2025 Senior Design Project

Gamified Habit Tracker: Leveling Up Everyday Routines

Students: Bryan A. Fernandez, Abhiram Bhogi, Juan Lao Romero, Shamar Weekes, George Ulloa
Instructor/Faculty: Masoud Sadjadi, Florida International University

PROBLEM

Most people struggle to maintain positive habits because traditional tracking apps lack the psychological reinforcement needed for long-term motivation.

Without immediate rewards or visible progress, users abandon their goals within weeks. Gamified Habit Tracker applies proven behavioral psychology—positive reinforcement, progressive goals, and social accountability—through RPG mechanics.

Users earn XP across five life attributes (Strength, Intelligence, Creativity, Social, Health), unlock achievements, and compete on leaderboards. AI-powered difficulty assessment ensures fair rewards, creating the dopamine-driven feedback loops that sustain behavioral change.

The result: users build sustainable routines that genuinely improve their physical health, mental well-being, creativity, and social connections—transforming tedious habit-tracking into an engaging journey of self-improvement.

VERIFICATION

- Unit tests validate core game logic including XP calculation, level progression, and achievement unlock conditions.
- Integration tests verify API endpoints for habit completion, XP awards, and leaderboard ranking updates.
- User acceptance testing ensures the complete habit-tracking workflow and gamification features function as intended.
- Performance testing measures API response times and database query efficiency to meet the 500ms requirement.
- Automated test suites run after each code change to catch regressions, while manual testing validates cross-platform compatibility.
- Load testing simulates concurrent users logging habits to verify system stability and database consistency.
- Security testing validates token authentication, password encryption, and user data protection measures.

SYSTEM DESIGN

- **Tech Stack:** Django REST API + React frontend with token-based auth and RESTful design.
- **AI Integration:** OpenAI GPT-4o-mini automatically calculates XP rewards based on habit difficulty analysis.
- **Core Mechanic:** Habit completion earns XP across 5 attributes (Strength, Intelligence, Creativity, Social, Health).
- **Gamification:** 21 achievements, character progression (3 unlockable characters), equipment system, and tower combat mini-game.
- **Social & Persistence:** Global leaderboard, daily check-ins, SQLite database with production scalability.



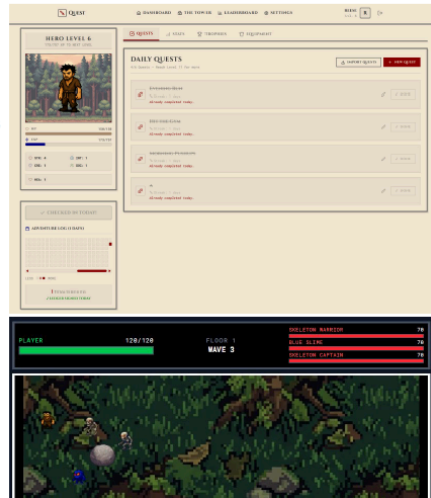
SOLUTION

Our capstone project, Gamified Habit Tracker, tackles the motivation crisis in habit-building by transforming daily routines into an engaging RPG adventure. We leverage positive psychological reinforcement—the same dopamine-driven mechanics that make games addictive—to help users build genuinely healthy, consistent habits that improve their lives.

The app turns habits into "quests" where users earn XP, level up across 5 life-improving attributes (Strength, Intelligence, Creativity, Social, Health), unlock achievements, and compete on leaderboards. AI-powered difficulty assessment using OpenAI's GPT-4o-mini ensures every accomplishment feels fairly rewarded, sustaining motivation over months rather than days.

Built with a Django REST API and React frontend, the system provides daily check-ins for consistency, character progression that visualizes personal growth, equipment unlocks that reward milestones, and real-time tower combat that makes habit streaks exciting. The result: users don't just track habits—they actively look forward to completing them, developing sustainable routines that enhance their fitness, learning, creativity, relationships, and overall well-being.

USER INTERFACE



REQUIREMENTS

Functional Requirements:

- Users can create, edit, and delete habits with customizable XP rewards.
- Users can log habit completions and view streaks.
- System automatically calculates XP gains and level progression.
- Users can unlock and equip character customization items.
- System tracks daily check-ins and awards 100 XP.
- Users can view character stats, achievements, and equipment.
- Global leaderboard displays users ranked by level and XP.
- Achievement system automatically unlocks badges at milestones.
- Admin panel manages users, habits, and achievements.
- User authentication via token-based login.

Non-Functional Requirements:

- Real-time XP and stat synchronization across frontend and backend.
- API response time under 500ms. for habit operations.
- Responsive UI on desktop and mobile devices.
- Support 100+ concurrent users without degradation.
- All user data encrypted and securely stored.
- 99% system uptime for habit tracking.
- Code follows Django and React best practices.

F. Scrum Evidence Index (links to Sprint Planning, Dailies, Reviews, Retros)

Sprint Planning:

- [≡ Sprint Planning Meeting Minutes - 11/24/25](#)
- [≡ Sprint Planning Meeting Minutes - 11/10/25](#)
- [≡ Sprint Planning Meeting Minutes - 10/27/25](#)
- [≡ Sprint Planning Meeting Minutes - 10/13/25](#)
- [≡ Sprint Planning Meeting Minutes - 9/29/25](#)
- [≡ Sprint Planning Meeting Minutes - 9/15/2025](#)
- [≡ Capstone 2 - Sprint Planing Meeting - 9/2/25](#)

Sprint Review :

- [≡ Capstone 2 - Sprint Review Meeting Minutes - 11/23/25](#)
- [≡ Capstone 2 - Sprint Review Meeting Minutes - 11/09/25](#)
- [≡ Capstone 2 - Sprint Review Meeting Minutes - 10/26/25](#)
- [≡ Capstone 2 - Sprint Review Meeting Minutes - 10/12/25](#)
- [≡ Capstone 2 - Sprint Review Meeting Minutes - 9/28/25](#)
- [≡ Capstone 2 - Sprint Review Meeting Minutes - 9/28/25](#)

Retros:

- [≡ Sprint Retrospective Meeting Minutes - 11/23/2025](#)
- [≡ Sprint Retrospective Meeting Minutes - 11/09/2025](#)
- [≡ Sprint Retrospective Meeting Minutes - 10/26/2025](#)
- [≡ Sprint Retrospective Meeting Minutes - 10/12/2025](#)
- [≡ Sprint Retrospective Meeting Minutes - 9/28/2025](#)
- [≡ Capstone 2 - Sprint Retrospective Meeting Minutes - 9/14/25](#)
- [≡ Capstone 2 - Sprint Retrospective Meeting Minutes - 9/14/25](#)

Dalies:

- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 12/1/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/28/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/27/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/26/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/25/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/24/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/21/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/20/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/19/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/18/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/17/25](#)
- [≡ Capstone 2 - Daily Scrum Meeting Minutes - 11/14/25](#)

- Capstone 2 - Daily Scrum Meeting Minutes - 11/13/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/12/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/11/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/10/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/7/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/6/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/5/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/4/25
- Capstone 2 - Daily Scrum Meeting Minutes - 11/3/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/31/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/30/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/29/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/28/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/27/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/24/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/23/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/22/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/21/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/20/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/17/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/16/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/15/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/14/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/13/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/10/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/9/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/8/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/7/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/6/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/03/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/2/25
- Capstone 2 - Daily Scrum Meeting Minutes - 10/1/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/30/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/29/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/26/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/25/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/24/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/23/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/22/25
- Capstone 2 - Daily Scrum Meeting Minutes - 9/19/25

- ☰ Capstone 2 - Daily Scrum Meeting Minutes - 9/18/25
- ☰ Capstone 2 - Daily Scrum Meeting Minutes - 9/17/25
- ☰ Capstone 2 - Daily Scrum Meeting Minutes - 9/16/25
- ☰ Capstone 2 - Daily Scrum Meeting Minutes - 9/15/2025
- ☰ Capstone 2 - Daily Standup - 9/12/25
- ☰ Capstone 2 - Daily Standup - 9/11/25
- ☰ Capstone 2 - Daily Standup - 9/10/25
- ☰ Capstone 2 - Daily Standup - 9/9/25
- ☰ Capstone 2 - Daily Standup - 9/8/25
- ☰ Capstone 2 - Daily Standup - 9/5/25
- ☰ Capstone 2 - Daily Standup - 9/4/25
- ☰ Capstone 2 - Daily Standup - 9/3/25
- ☰ Capstone 2 - Daily Standup - 9/2/25