ITE 18 – Application Development and Emerging Technologies

Web Application - Final Project Submission

Student Name(s): Van Renfred M. Otacan

Project Title: Enderchive - Game Collection Management System

Date of Submission: 12/26/25

Course/Section: EJ1

---

1. Project Overview

1.1 Project Description

Enderchive is a comprehensive web application designed for gamers to manage, track, and share their video game collections. The application solves the problem of disorganized game libraries by providing users with a centralized platform to catalog their games, track their progress, write reviews, and connect with friends who share similar gaming interests.

The application allows users to:

- Build and organize their personal game library with detailed information
- Track game progress through milestones and playtime
- Rate and review games
- Maintain a wishlist of desired games
- Manage game ownership across different platforms and editions
- Connect with friends and view their gaming activities
- Discover new games through search and browsing features

1.2 Target Users

The primary target users of Enderchive are:

1. **Casual Gamers**: Individuals who play games occasionally and want to keep track of their collection and gaming history
2. **Hardcore Gamers**: Enthusiasts who own extensive game libraries across multiple platforms and want detailed tracking and organization
3. **Game Collectors**: Users who collect physical and digital game editions and need a system to catalog their collection
4. **Social Gamers**: Players who want to share their gaming experiences, reviews, and recommendations with friends
5. **Gaming Communities**: Groups of friends who want to compare libraries, share recommendations, and discuss games

User Characteristics:

- Age range: 16-45 years old
- Tech-savvy individuals comfortable with web applications
- Active gamers across various platforms (PC, Console, Mobile)
- Users seeking organization and social interaction around gaming

1.3 Key Features

- **User Authentication & Profile Management**: Secure registration, login, and profile customization with username, bio, and avatar support
- **Game Library Management**: Add games to personal library with status tracking (Wishlist, Playing, Completed, Abandoned)
- **Game Catalog**: Comprehensive game database with details including title, description, genre, developer, publisher, release date, platforms, and tags
- **Progress Tracking**: Track game progress through milestones, playtime hours, and completion status
- **Game Ownership Management**: Record ownership of different game editions across various stores and platforms
- **Review System**: Write and manage game reviews with ratings (1-5 scale) and text reviews
- **Wishlist**: Maintain a wishlist of games users want to acquire
- **Friend System**: Send and manage friend requests, view friends' libraries and activities
- **Search & Discovery**: Search games by title, filter by genre, platform, and tags
- **Admin Dashboard**: Administrative interface for managing users, games, genres, platforms, and tags
- **Responsive Design**: Fully responsive interface that works on desktop, tablet, and mobile devices

1.4 Technology Stack

Frontend:

- Vue.js 3.5.26 (Progressive JavaScript Framework)
- Inertia.js 2.3.4 (Server-driven Single Page Applications)
- Tailwind CSS 4.0.0 (Utility-first CSS Framework)
- Lucide Vue Next 0.562.0 (Icon Library)
- Axios 1.11.0 (HTTP Client)
- Vite 7.0.7 (Build Tool)

Backend:

- Laravel 12.0 (PHP Web Framework)
- PHP 8.2+ (Server-side Language)
- Laravel Sanctum 4.0 (API Authentication)

Database:

- MySQL 8.0+ (Primary database)

Other Tools:

- Git (Version Control)
- Composer (PHP Dependency Manager)
- NPM (Node Package Manager)
- Laravel Pint (Code Style Fixer)
- PHPUnit (Testing Framework)

---

2. App Plan

2.1 Project Scope

In Scope:

- User registration, authentication, and profile management
- Game catalog with CRUD operations
- Personal game library management (add, update, delete games)
- Game status tracking (Wishlist, Playing, Completed, Abandoned)
- Game progress tracking with milestones and playtime
- Game ownership management (editions, stores, platforms)
- Review and rating system
- Friend request and management system
- Search and filter functionality
- Admin dashboard for content management
- Responsive web interface
- API endpoints for all features
- Input validation and error handling
- Security measures (authentication, authorization, CSRF protection)

Out of Scope:

- Real-time notifications (push notifications)
- Social media integration
- Payment processing
- Game purchase functionality
- Mobile native applications (iOS/Android)
- Multi-language support

- Game achievement synchronization from external APIs
- Cloud storage for game images
- Email notifications
- Password reset functionality (can be added in future)

2.2 Objectives & Goals

1. **Primary Objective**: Develop a fully functional web application that allows users to manage their game collections with comprehensive tracking features

2. **User Experience Objective**: Create an intuitive, responsive interface that provides seamless navigation and user feedback

3. **Data Management Objective**: Implement a normalized database structure that efficiently stores and retrieves game and user data

4. **Security Objective**: Ensure secure authentication, authorization, and data protection following Laravel best practices

5. **Code Quality Objective**: Maintain clean, well-organized code following MVC architecture and industry standards

6. **Feature Completeness Objective**: Deliver all core CRUD operations for games, user games, reviews, and friend management

2.3 User Stories & Use Cases

User Story 1: User Registration and Authentication

- As a new user, I want to register an account with my email, username, and password, so that I can access the application and start building my game library
- Acceptance Criteria:

User Story 2: Add Game to Library

- As a user, I want to add games to my personal library and set their status, so that I can track which games I own, want, or have completed
- Acceptance Criteria:

User Story 3: Track Game Progress

- As a user, I want to mark game milestones as completed and track my playtime, so that I can monitor my progress through games
- Acceptance Criteria:

User Story 4: Write Game Reviews

- As a user, I want to write and edit reviews for games I've played, so that I can share my opinions and help others discover games
- Acceptance Criteria:

User Story 5: Manage Friends

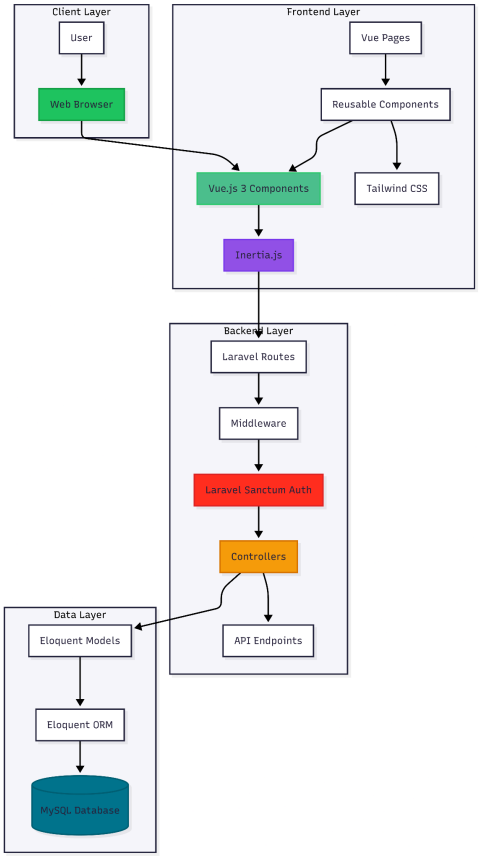
- As a user, I want to send friend requests and manage my friend list, so that I can view my friends' game libraries and activities
- Acceptance Criteria:

User Story 6: Search and Discover Games

- As a user, I want to search for games by title and filter by genre, platform, and tags, so that I can find games to add to my library
- Acceptance Criteria:

2.4 System Architecture

Enderchive follows a modern full-stack architecture using Laravel as the backend API and Vue.js with Inertia.js for the frontend:



Architecture Overview:

1. **Frontend Layer (Vue.js + Inertia.js)**
2. **Backend Layer (Laravel)**
3. **Database Layer**
4. **Authentication Layer**

Data Flow:

- User interacts with Vue.js frontend components
- Inertia.js sends requests to Laravel routes
- Laravel controllers process requests, validate input, and interact with models
- Eloquent ORM queries the database
- Data is returned to frontend via Inertia responses or JSON API responses
- Vue.js components update the UI reactively

2.5 Development Timeline

Phase	**Description**	**Timeline**
Phase 1	Design & Planning (ERD, UI/UX Design, App Plan)	Week 1-2
Phase 2	Database Setup & Migrations	Week 2-3
Phase 3	Backend Development (Models, Controllers, API Routes)	Week 3-5
Phase 4	Frontend Development (Vue Components, Pages, Styling)	Week 4-7
Phase 5	Integration, Testing & Bug Fixes	Week 7-8
Phase 6	Documentation & Final Polish	Week 8

---

3. UI/UX Design

3.1 Design Philosophy

The design philosophy of Enderchive centers around creating an immersive gaming experience that feels modern, intuitive, and visually appealing. The application follows key UI/UX principles:

Principles Applied:

1. **Usability & Accessibility**: Clear navigation structure, readable text sizes, sufficient color contrast, and logical information hierarchy ensure users can easily find and interact with features

2. **Consistency**: Uniform button styles, color schemes, typography, spacing, and interaction patterns throughout the application create a cohesive user experience

3. **Visual Hierarchy**: Important information stands out through strategic use of size, color, contrast, and spacing, guiding users' attention naturally

4. **Feedback & Responsiveness**: Visual feedback for all user actions including loading indicators, success/error messages, and button state changes

5. **Simplicity & Minimalism**: Clean interface design that avoids clutter, with every element serving a clear purpose

6. **User-Centered Design**: Features designed based on actual user needs for game collection management and social interaction

7. **Learnability**: Familiar patterns and conventions (standard navigation placement, recognizable icons) help users understand the interface quickly

8. **Aesthetic Appeal**: Dark theme with vibrant accent colors creates an engaging gaming aesthetic that builds trust and encourages engagement

3.2 Color Scheme

The application uses a dark gaming-themed color palette:

- **Primary Color:** Green (#22c55e) - Used for brand identity, primary actions, and highlights. Represents gaming and energy
- **Secondary Color:** Purple/Dark Purple (rgba(94,0,160,0.35)) - Used for borders, accents, and gradient backgrounds. Creates depth and gaming atmosphere
- **Accent Color:** Green with glow effect (#22c55e with drop-shadow) - Used for interactive elements, icons, and emphasis
- **Background Color:** Dark gradient (rgba(10,0,21,0.9) to rgba(30,0,62,0.85)) - Creates immersive dark theme suitable for gaming content
- **Text Color:** White/Light Gray - Ensures readability against dark backgrounds
- **Card Background:** Semi-transparent dark (rgba(27, 22, 38, 0.6)) - Provides depth while maintaining readability

3.3 Typography

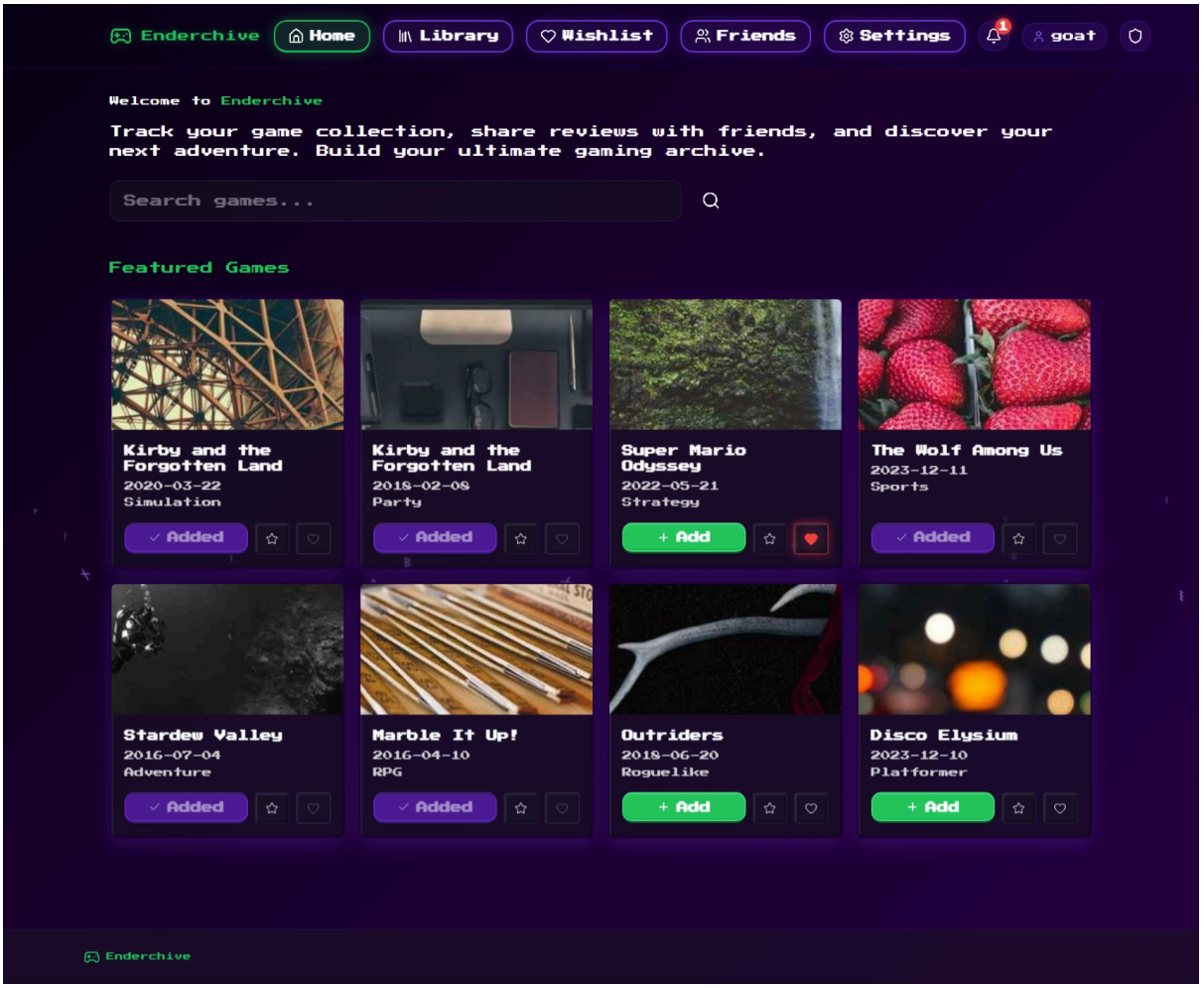
- **Font Family:** System fonts (sans-serif stack) - Ensures fast loading and native feel across devices
- **Heading Font Size:** 14px-24px (responsive) - Clear hierarchy for section titles and page headers
- **Body Font Size:** 14px-16px (responsive) - Optimal readability for content
- **Line Height:** 1.5 - Comfortable reading spacing
- **Letter Spacing:** 0.05em for brand text - Adds distinctive character to logo/branding
- **Text Shadow:** Applied to brand text for visual emphasis and glow effects

3.4 UI Components

- **Button (GameButton):** Gradient-styled buttons with hover effects, active states, and disabled states. Used for primary actions throughout the application
- **Input Fields (GameInput):** Styled input fields with dark theme, proper focus states, and validation feedback. Used for search, forms, and data entry

- **Navigation Bar:** Sticky header with gradient background, blur effect, and responsive design. Contains logo, navigation tabs, and user actions
- **Cards (GameItemCard):** Game display cards with cover images, title, platform badges, and action buttons. Hover effects for interactivity
- **Modal/Dialog:** Used for confirmations, forms, and detailed views. Includes backdrop blur and smooth transitions
- **Navigation Tabs:** Tab-based navigation with active state indicators and icons. Responsive design with mobile menu alternative
- **Notification System:** Toast notifications for user feedback (success, error, info messages)
- **Loading States:** Spinner indicators and skeleton screens during data fetching
- **Status Badges:** Visual indicators for game status (Wishlist, Playing, Completed, Abandoned) with distinct colors

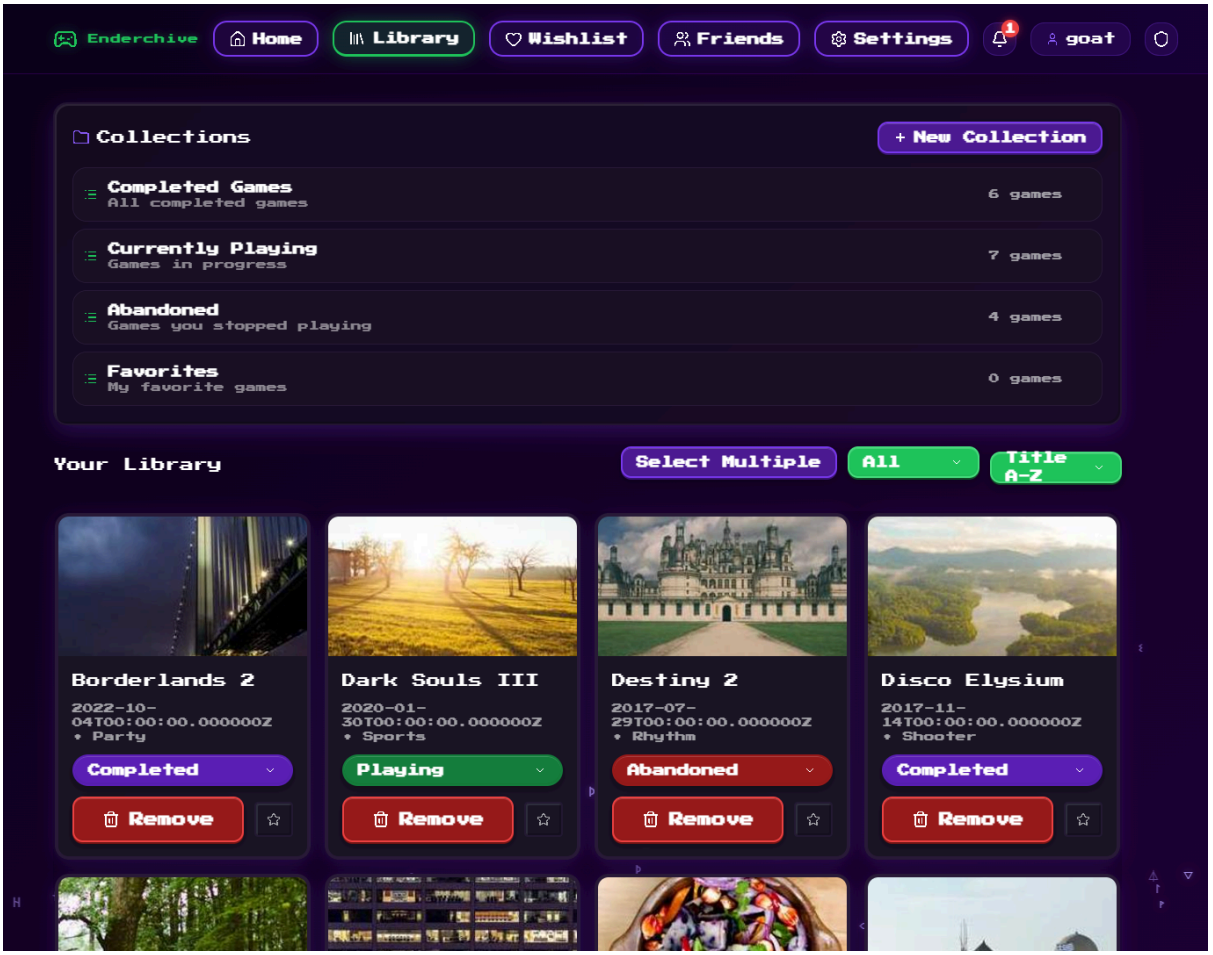
3.5 Wireframes & Mockups



Screen 1: Home Page

Description: The home page serves as the main landing and discovery page. Key elements include:

- Sticky navigation bar with logo, main navigation tabs (Home, Library, Wishlist, Friends, Settings), and user profile/notification icons
- Welcome section with application description
- Search bar for finding games
- Featured games section displaying popular or recent games
- Responsive layout that adapts to different screen sizes



Screen 2: Library Page

Description: The library page displays the user's personal game collection. Key elements include:

- Filter options for game status (Wishlist, Playing, Completed, Abandoned)
- Grid/list view of game cards
- Each game card shows cover image, title, platform badges, status, rating, and quick actions
- Add game button to add new games to library
- Search and sort functionality within the library

Enderchive | Home | Library | Wishlist | Friends | Settings | goat

← Back

Kirby and the Forgotten Land

3/22/2020 • Simulation

Itaque blanditiis et ut. Possimus sed et qui molestias. Sed vitae aut aspernatur enim. Aut repudiandae eaque commodi amet neque. Asperiores expedita tenetur dolor ducimus ut in sit possimus.

Quos totam labore aut quo. Excepturi ipsus et exercitationem similique tempore commodi. Autem a itaque fugit magni quod.

Reiciendis et exercitationem voluptatem quia. Quaerat expedita a eum nulla est. Et et tenetur consequatur neque illum exercitationem.

Progress

Completion 0%

Playtime 0h

Achievements 0/6

Game Journal + Add Entry

No journal entries yet

Document your gaming journey

Your Copies

Nintendo Switch • Owned • Digital • PlayStation Store
Account: user_4795
Purchased Jun 27, 2024 USD 50.69

PlayStation 5 • Owned • Physical • Xbox Store
Account: user_3795
Purchased Jul 27, 2029

PlayStation Vita • Owned • Physical • Nintendo eShop
Game of the Year Edition
Account: user_4405
Purchased Jun 10, 2029

Nintendo DS • Gifted • Digital • Epic Games Store

+ Add Copy

Rating ♥♥♥♡♡ 3.1/5

Genres Simulation

Released 3/22/2020

Platforms Xbox One, PC, Android

♡ Add to Wishlist

Remove from Library

Game Achievements

0 / 6 unlocked

+ Complete Achievement

No achievements completed yet

Complete your first achievement to track your progress!

Available Achievements (6)

Collect All Items (100 pts) · Defeat Final Boss (200 pts) · Defeat First Boss (50 pts) · Complete Main Story (300 pts) · Reach Mid-Game (100 pts) · Complete Tutorial (10 pts)

Proof Gallery

No proofs uploaded yet

Reviews

7 reviews

✎ Write Review

verdenef
★★★★★ 12/25/2025
cute game iliv

Keegan Rice
★★★★★ 10/29/2025
The difficulty is balanced. Eum et iure aut ut. Nesciunt unde qui sint voluptatem. Voluptas quibusdam distinctio itaque velit nemo. Aut repellat quia sed sed. Asperiores deleniti molestias animi omnis et ut perferendis. Nihil molestiae culpa facere hic eius numquam consequuntur est. Voluptatum dicta iure expedita. Voluptates ut sapiente molestiae quod labore sit nihil. Possimus veritatis eveniet ut blanditiis officiis.

Ms. Abby Schinner
★★★★★ 10/29/2025
Single player mode is satisfying. Quod cupiditate mollitia reiciendis a non. Non ullam et doloribus. Minus ipsam assumenda sint earum fugit et excepturi. Nihil hic occaecati et qui vel quo.

Oswaldo Beier II
★★★★★ 10/29/2025
The difficulty is accessible. Quidem voluptas minima consequatur omnis. Quis repudiandae sed nesciunt veritatis illo necessitatibus. Maxime quos sit ab et ipsam quas. Vel aliquam est voluptatem fuga illum.

Nannie Dare
★★★★★ 10/29/2025
Controls feel fluid. Sit voluptatem voluptatem reprehenderit error rerum dolore natus possimus. Tenetur quo pariatur possimus dolores qui. Mollitia animi et unde est sed velit molestiae. Repellat numquam ullam ad expedita voluptatem vel. Modi debitis aut dolores non explicabo est. Beatae non totam esse et voluptatem totam. Deserunt eaque non eos in et ex. Doloribus sint sunt et culpa ut sit. Sunt quisquam est et tempore voluptatem et eius. Saepe rerum laudantium eius eum ipsam. Alias mollitia ad nihil enim. Eius praesentium voluptas porro harum veniam ea est.

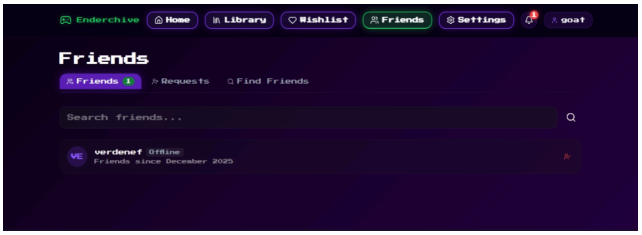
Leslie Wilderman
★★★★★ 10/29/2025
Multiplayer experience is competitive. Cupiditate et non voluptas sed id et enim. Aut aperiam et accusantium ipsam placeat sapiente voluptas. Molestiae dolorum quo voluptatem voluptatibus esse. Maxime quo corporis ipsum tempore fugit perspiciatis sequi.

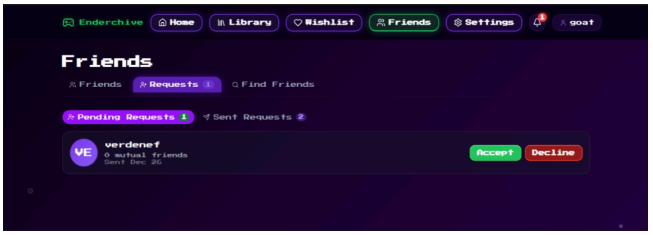
Reba Feest
★★★☆☆ 10/29/2025
Graphics are detailed. Est architecto aut ut animi aliam officia. Provident et sunt et. Temporibus nostrum qui exercitationes aut eligendi dolorem consequatur quo. Ea qui esse facere quibusdam. Fugit modi commodi id sed iure sunt reiciendis. Est reiciendis eum aut vel quae perspiciatis nobis. Vel molestiae inventore voluptates illum quibusdam qui. Exercitationem non hic rerum deleniti laudantium rerum enim eligendi. Sed non deleniti at. Eos repellendus dolore ut aut alias. Doloribus ut ullam dignissimos dolor aut dolores. Nulla maxime atque molestias totam blanditiis qui.

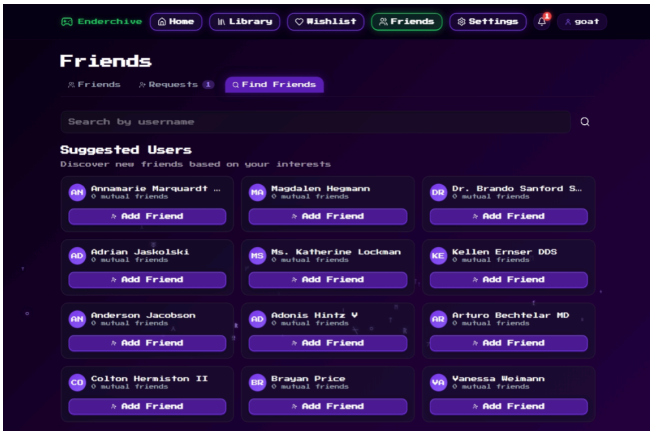
Screen 3: Game Detail Page

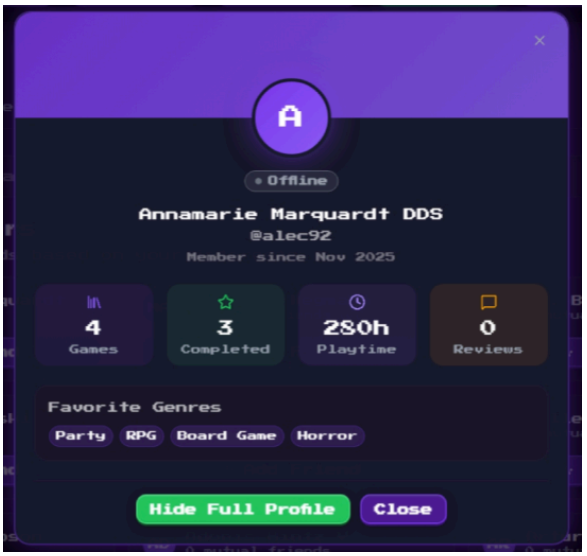
Description: The game detail page provides comprehensive information about a specific game. Key elements include:

- Large game cover image and title
- Game information (description, genre, developer, publisher, release date, platforms, tags)
- User's personal game data (status, rating, playtime, progress)
- Milestones section with progress tracking
- Reviews section showing user and friend reviews
- Ownership management (editions, stores)
- Action buttons (add to library, update status, write review)







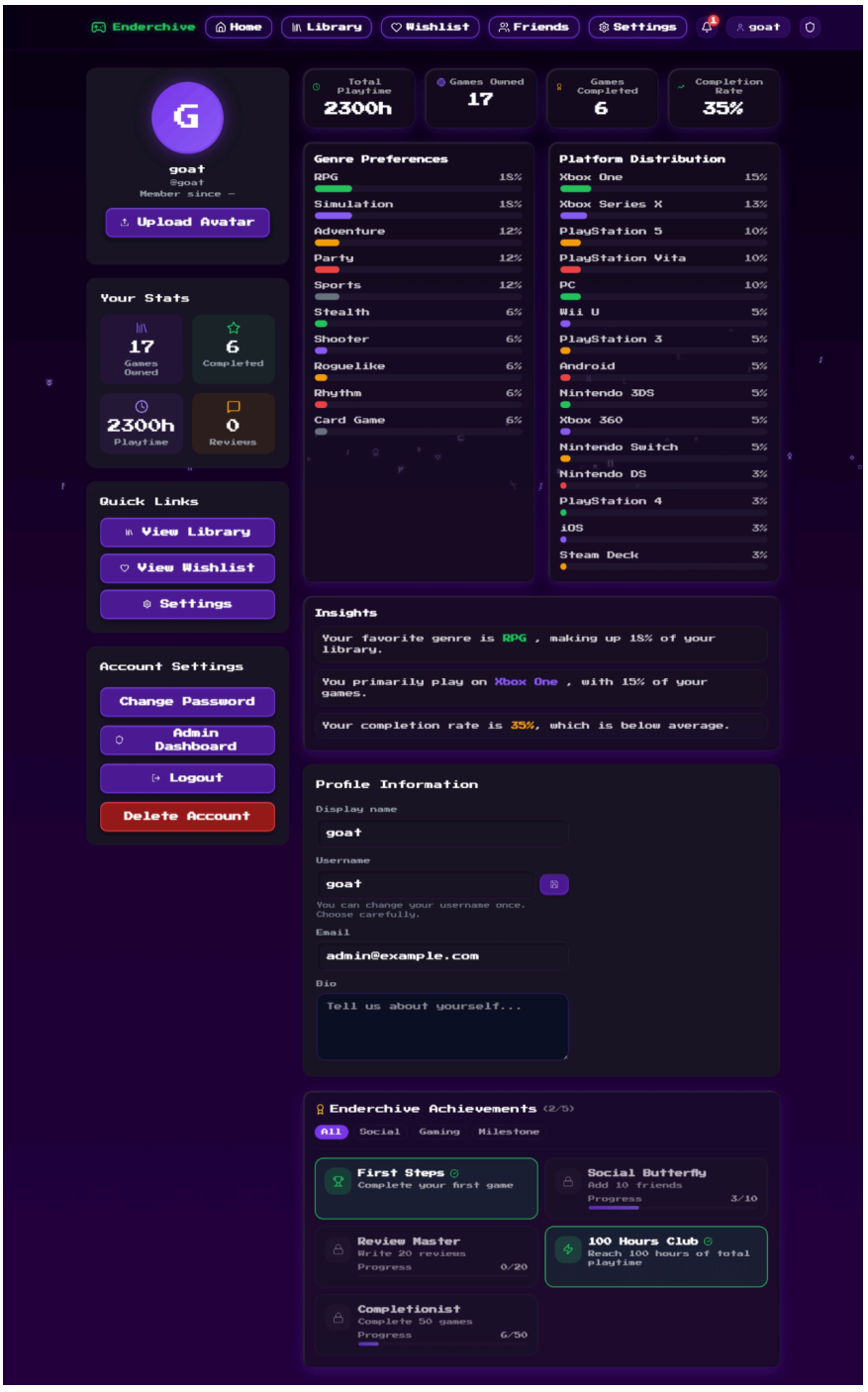


Screen 4: Friends Page

Description: The friends page manages social connections. Key elements include:

- Tab navigation (Friends, Requests, Search)
- Friends list with avatars and usernames

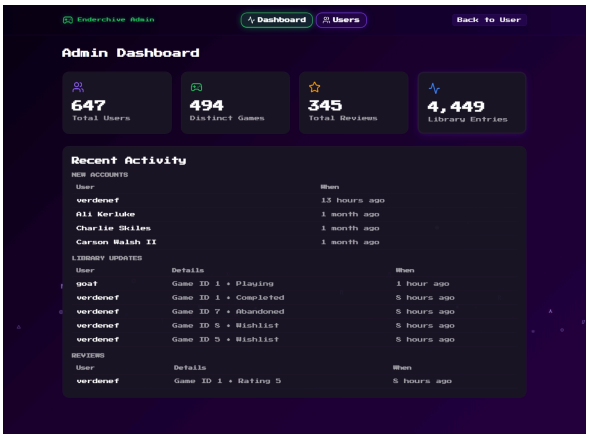
- Friend request notifications
- Search functionality to find users
- Friend profile cards showing their game statistics
- Action buttons (send request, accept/decline, remove friend)

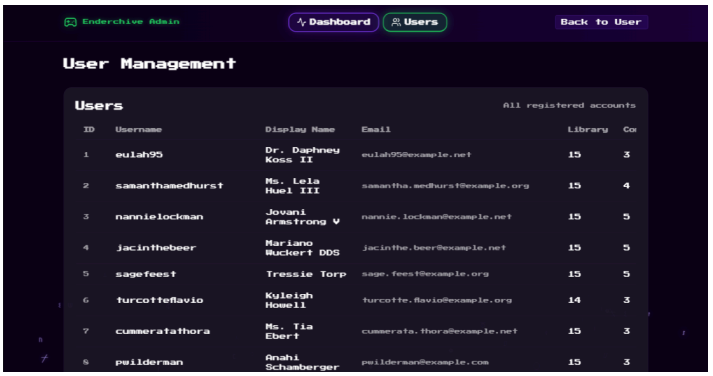


Screen 5: Profile Page

Description: The profile page allows users to manage their account. Key elements include:

- Profile information editing (name, username, bio, avatar)
- Password change functionality
- Account preferences
- Form validation and error messages
- Save/cancel actions





Screen 6: Admin Dashboard

Description: The admin dashboard provides administrative controls. Key elements include:

- Summary statistics (total users, games, reviews)
- User management interface
- Content management (games, genres, platforms, tags)
- Activity monitoring
- Tab-based navigation for different admin functions

3.6 User Flows

Flow 1: User Registration and First Game Addition

1. User visits the application homepage
2. User clicks "Sign Up" or navigates to registration page
3. User fills in registration form (name, username, email, password, password confirmation)
4. System validates input (unique email/username, password strength, matching passwords)
5. User account is created and user is automatically logged in
6. User is redirected to home page
7. User searches for a game using the search bar
8. User clicks on a game from search results
9. User views game details page
10. User clicks "Add to Library" button

11. User selects game status (Wishlist, Playing, Completed, or Abandoned)
12. User optionally sets rating and playtime
13. Game is added to user's library
14. User is redirected to library page where the game appears

Flow 2: Writing and Managing Game Reviews

1. User navigates to a game detail page (from library or search)
2. User scrolls to reviews section
3. User clicks "Write Review" button
4. User fills in review form (rating 1-5, optional review text)
5. User submits review
6. System validates input and saves review
7. Review appears in reviews section
8. User can later edit review by clicking "Edit" on their review
9. User modifies rating or text and saves changes
10. Updated review is displayed
11. User can delete review by clicking "Delete" button and confirming

Flow 3: Friend Request and Library Viewing

1. User navigates to Friends page
2. User clicks on "Search" tab
3. User enters username or name in search field
4. System displays matching users
5. User clicks "Send Friend Request" on desired user
6. Friend request is sent and status changes to "Pending"
7. Target user receives notification (if implemented) or sees request in "Requests" tab
8. Target user navigates to Friends page and "Requests" tab
9. Target user sees pending friend request
10. Target user clicks "Accept" button
11. Friend relationship is established (status changes to "Accepted")
12. Both users can now view each other's profiles and libraries
13. User navigates to friend's profile to view their game collection

Flow 4: Tracking Game Progress with Milestones

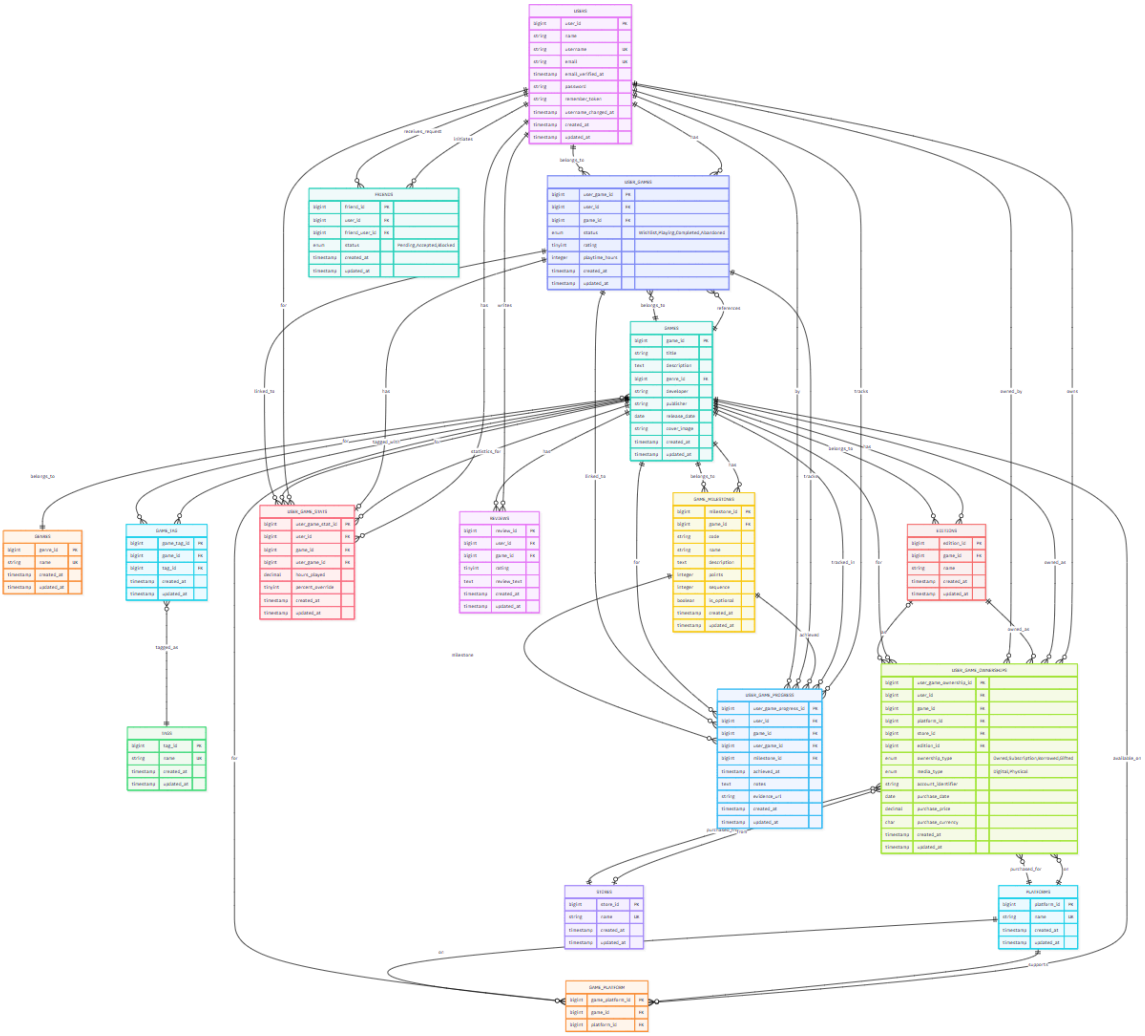
1. User navigates to their library
2. User clicks on a game with status "Playing"
3. User views game detail page

4. User scrolls to milestones section
5. User views available milestones for the game
6. User clicks "Mark Complete" on a milestone they've achieved
7. System records milestone completion with timestamp
8. Progress indicator updates to show completed milestones
9. User can update total playtime hours
10. User can mark game as "Completed" when all milestones are done
11. Progress data is saved and displayed in user's game statistics

---

4. Database Architecture (ERD)

4.1 Entity Relationship Diagram



4.2 Entity Descriptions

Entity 1: users

Field	**Data Type**	**Constraints**	**Description**
user_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each user
name	VARCHAR(255)	NOT NULL	User's full name
username	VARCHAR(50)	UNIQUE, NULLABLE	Unique username for user identification
email	VARCHAR(255)	UNIQUE, NOT NULL	User's email address (used for login)
email_verified_at	TIMESTAMP	NULLABLE	Timestamp when email was verified
password	VARCHAR(255)	NOT NULL	Hashed password for authentication
remember_token	VARCHAR(100)	NULLABLE	Token for "remember me" functionality
bio	TEXT	NULLABLE	User's biography/description
avatar_url	VARCHAR(255)	NULLABLE	URL to user's avatar image
username_changed_at	TIMESTAMP	NULLABLE	Timestamp of last username change
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 2: games

Field	**Data Type**	**Constraints**	**Description**
game_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each game
title	VARCHAR(255)	NOT NULL	Game title
description	TEXT	NULLABLE	Game description/synopsis
genre_id	BIGINT	FOREIGN KEY, NULLABLE	Reference to genres table
developer	VARCHAR(255)	NULLABLE	Game developer name
publisher	VARCHAR(255)	NULLABLE	Game publisher name

Field	**Data Type**	**Constraints**	**Description**
release_date	DATE	NULLABLE	Game release date
cover_image	VARCHAR(255)	NULLABLE	URL to game cover image
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 3: user_games

Field	**Data Type**	**Constraints**	**Description**
user_game_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for user-game relationship
user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
status	ENUM	NOT NULL, DEFAULT 'Wishlist'	Game status: 'Wishlist', 'Playing', 'Completed', 'Abandoned'
rating	TINYINT	NULLABLE	User's rating (1-5 scale)
playtime_hours	INTEGER	DEFAULT 0	Total hours played
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 4: reviews

Field	**Data Type**	**Constraints**	**Description**
review_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each review
user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
rating	TINYINT	NULLABLE	Review rating (1-5 scale)
review_text	TEXT	NULLABLE	Review text content
created_at	TIMESTAMP	NULLABLE	Record creation timestamp

Field	**Data Type**	**Constraints**	**Description**
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 5: friends

Field	**Data Type**	**Constraints**	**Description**
friend_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for friend relationship
user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table (initiator)
friend_user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table (target)
status	ENUM	NOT NULL, DEFAULT 'Pending'	Friend status: 'Pending', 'Accepted', 'Blocked'
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 6: platforms

Field	**Data Type**	**Constraints**	**Description**
platform_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each platform
name	VARCHAR(255)	NOT NULL, UNIQUE	Platform name (e.g., PC, PlayStation, Xbox)
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 7: genres

Field	**Data Type**	**Constraints**	**Description**
genre_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each genre
name	VARCHAR(255)	NOT NULL, UNIQUE	Genre name (e.g., Action, RPG, Strategy)
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 8: tags

Field	**Data Type**	**Constraints**	**Description**
tag_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each tag
name	VARCHAR(255)	NOT NULL, UNIQUE	Tag name (e.g., Multiplayer, Single-player, Indie)
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 9: game_platform (Pivot Table)

Field	**Data Type**	**Constraints**	**Description**
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
platform_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to platforms table

Entity 10: game_tag (Pivot Table)

Field	**Data Type**	**Constraints**	**Description**
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
tag_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to tags table

Entity 11: editions

Field	**Data Type**	**Constraints**	**Description**
edition_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each edition
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
name	VARCHAR(255)	NOT NULL	Edition name (e.g., Standard, Deluxe, Collector's)
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 12: stores

Field	**Data Type**	**Constraints**	**Description**
store_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each store
name	VARCHAR(255)	NOT NULL, UNIQUE	Store name (e.g., Steam, Epic Games, GOG)
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 13: game_milestones

Field	**Data Type**	**Constraints**	**Description**
milestone_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each milestone
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
name	VARCHAR(255)	NOT NULL	Milestone name (e.g., "Complete Chapter 1")
description	TEXT	NULLABLE	Milestone description
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 14: user_game_ownerships

Field	**Data Type**	**Constraints**	**Description**
ownership_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each ownership record
user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
edition_id	BIGINT	FOREIGN KEY, NULLABLE	Reference to editions table
store_id	BIGINT	FOREIGN KEY, NULLABLE	Reference to stores table
platform_id	BIGINT	FOREIGN KEY, NULLABLE	Reference to platforms table
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 15: user_game_progress

Field	**Data Type**	**Constraints**	**Description**
progress_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each progress record
user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
milestone_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to game_milestones table
completed_at	TIMESTAMP	NULLABLE	Timestamp when milestone was completed
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

Entity 16: user_game_stats

Field	**Data Type**	**Constraints**	**Description**
stat_id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each stat record
user_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to users table
game_id	BIGINT	FOREIGN KEY, NOT NULL	Reference to games table
stat_name	VARCHAR(255)	NOT NULL	Stat name (e.g., "Kills", "Deaths", "Score")
stat_value	VARCHAR(255)	NULLABLE	Stat value
created_at	TIMESTAMP	NULLABLE	Record creation timestamp
updated_at	TIMESTAMP	NULLABLE	Record last update timestamp

4.3 Relationships

Relationship 1: users ← One-to-Many → user_games

Description: A user can have many games in their library. Each user_game record belongs to one user and one game. This relationship allows users to maintain their personal game collection with individual status, rating, and playtime for each game.

Relationship 2: games ← One-to-Many → user_games

Description: A game can be owned by many users. Each user_game record represents one user's relationship with one game, allowing multiple users to track the same game independently.

Relationship 3: users ← One-to-Many → reviews

Description: A user can write multiple reviews for different games. Each review belongs to one user and one game, enabling users to share their opinions on games they've played.

Relationship 4: games ← One-to-Many → reviews

Description: A game can have multiple reviews from different users. This allows the community to share diverse opinions and ratings for each game.

Relationship 5: users ← One-to-Many → friends (as initiator)

Description: A user can send multiple friend requests. The user_id field represents the user who initiated the friend request.

Relationship 6: users ← One-to-Many → friends (as target)

Description: A user can receive multiple friend requests. The friend_user_id field represents the user who received the friend request.

Relationship 7: games ← Many-to-Many → platforms

Description: A game can be available on multiple platforms, and a platform can have many games. This many-to-many relationship is implemented through the game_platform pivot table, allowing games to be associated with multiple platforms (e.g., PC, PlayStation, Xbox).

Relationship 8: games ← Many-to-Many → tags

Description: A game can have multiple tags, and a tag can be associated with many games. This many-to-many relationship is implemented through the game_tag pivot table, enabling flexible categorization and filtering of games.

Relationship 9: genres ← One-to-Many → games

Description: A genre can have many games, but each game belongs to one primary genre. This allows games to be categorized by genre while maintaining a primary genre classification.

Relationship 10: games ← One-to-Many → game_milestones

Description: A game can have multiple milestones that users can complete. Each milestone belongs to one specific game and can be tracked by multiple users through the user_game_progress table.

Relationship 11: users ← One-to-Many → user_game_progress

Description: A user can track progress on multiple games and milestones. Each progress record represents a user completing a specific milestone for a specific game.

Relationship 12: games ← One-to-Many → user_game_progress

Description: A game can have progress tracked by multiple users. Each progress record links a user, game, and milestone together.

Relationship 13: users ← One-to-Many → user_game_ownerships

Description: A user can own multiple game editions across different stores and platforms. Each ownership record represents one specific ownership instance.

Relationship 14: games ← One-to-Many → editions

Description: A game can have multiple editions (Standard, Deluxe, Collector's, etc.). Each edition belongs to one game.

Relationship 15: games ← One-to-Many → user_game_ownerships

Description: A game can be owned by multiple users in different editions, stores, and platforms. Each ownership record represents one specific ownership instance.

4.4 Database Normalization

The database follows Third Normal Form (3NF) normalization principles:

First Normal Form (1NF):

- All tables have primary keys (user_id, game_id, etc.)
- All columns contain atomic values (no multi-valued attributes)

- No repeating groups within tables
- Each field contains a single value

Second Normal Form (2NF):

- All tables are in 1NF
- All non-key attributes are fully functionally dependent on the primary key
- Example: In the `user_games` table, status, rating, and playtime_hours are fully dependent on the composite key (user_id, game_id), not just on user_id or game_id alone

Third Normal Form (3NF):

- All tables are in 2NF
- No transitive dependencies exist (non-key attributes don't depend on other non-key attributes)
- Example: In the `games` table, genre information is stored in a separate `genres` table to avoid redundancy. The game table only stores genre_id as a foreign key reference.

Additional Normalization Considerations:

- Pivot tables (game_platform, game_tag) properly handle many-to-many relationships
- Foreign key constraints ensure referential integrity
- Proper indexing on foreign keys for performance
- Separation of concerns: user data, game data, and relationship data are stored in separate tables
- No data redundancy: information like platform names, genre names, and tag names are stored once and referenced

Benefits of Normalization:

- Data integrity: Prevents update anomalies and inconsistencies
- Storage efficiency: Eliminates redundant data storage
- Query flexibility: Allows complex queries across related tables
- Maintainability: Changes to data structure are easier to implement
- Scalability: Database can grow without significant performance degradation

---

5. Application Features & Functionality

5.1 Feature 1: User Authentication & Profile Management

Description: Secure user registration, login, and profile management system that allows users to create accounts, authenticate, and customize their profiles.

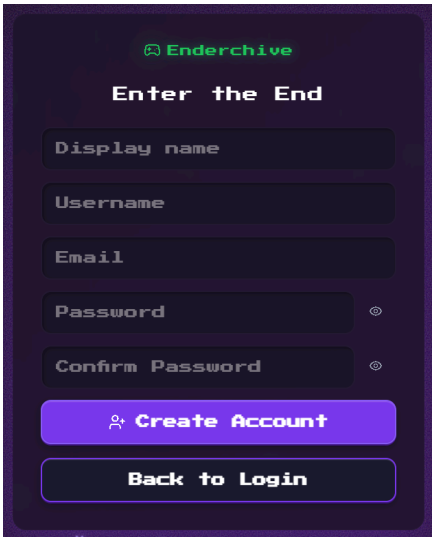
Functionality:

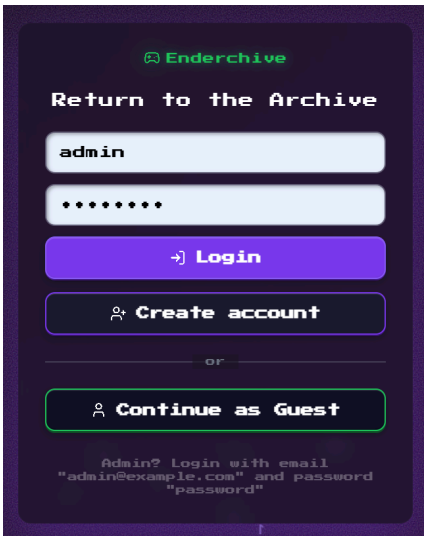
- User registration with email, username, name, and password
- Login with email or username
- Password hashing using bcrypt
- Session-based authentication for web routes
- API token authentication (Laravel Sanctum) for API routes
- Profile viewing and editing (name, username, bio, avatar)
- Username change tracking (username_changed_at timestamp)
- Password change functionality
- Remember me functionality
- Logout functionality

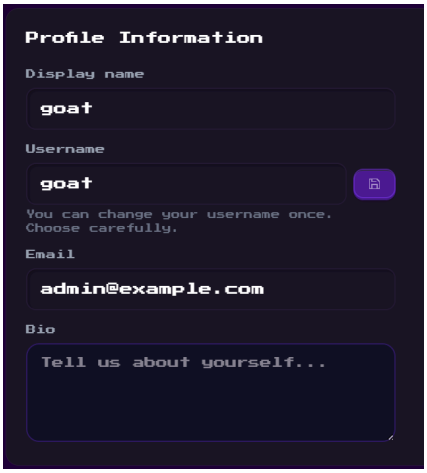
Implementation Details:

Authentication is implemented using Laravel's built-in authentication system with Sanctum for API tokens. The `AuthenticationController` handles registration and login, validating input using Laravel's Form Request validation. Passwords are automatically hashed using Laravel's Hash facade. Session-based authentication is used for web routes (Inertia.js), while Sanctum tokens are used for API routes. The User model uses Laravel's Authenticatable trait and includes relationships to user_games, reviews, and friends.

Screenshots:







5.2 Feature 2: Game Library Management

Description: Comprehensive system for users to add, organize, and manage their personal game collection with status tracking, ratings, and playtime.

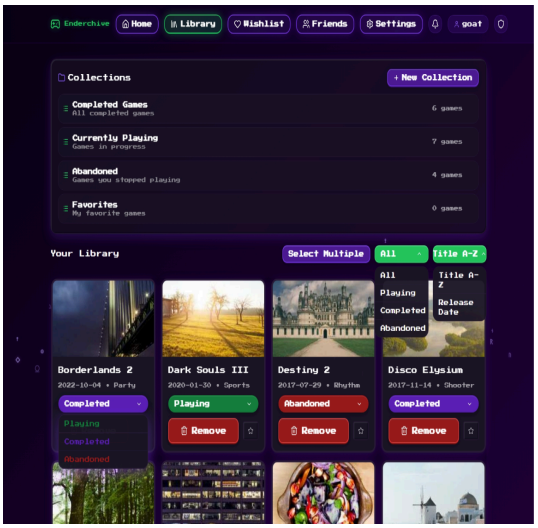
Functionality:

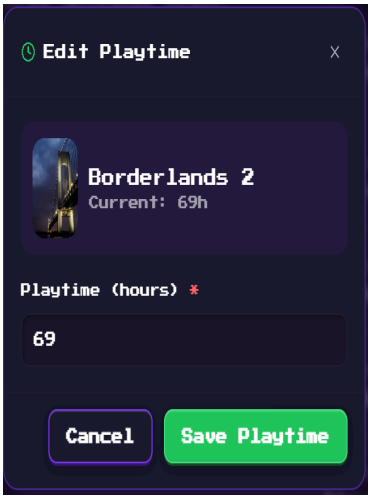
- Add games to personal library
- Set game status (Wishlist, Playing, Completed, Abandoned)
- Rate games on a 1-5 scale
- Track playtime in hours
- View library with filtering by status
- Update game status, rating, and playtime
- Remove games from library
- View library statistics

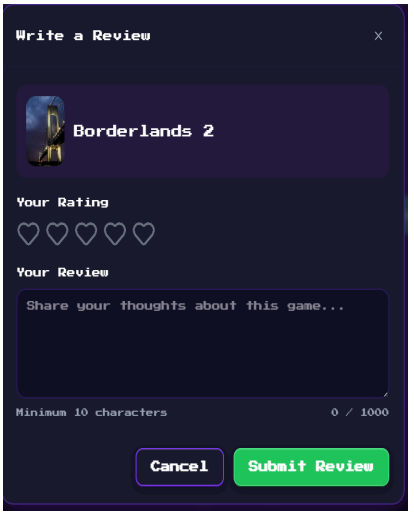
Implementation Details:

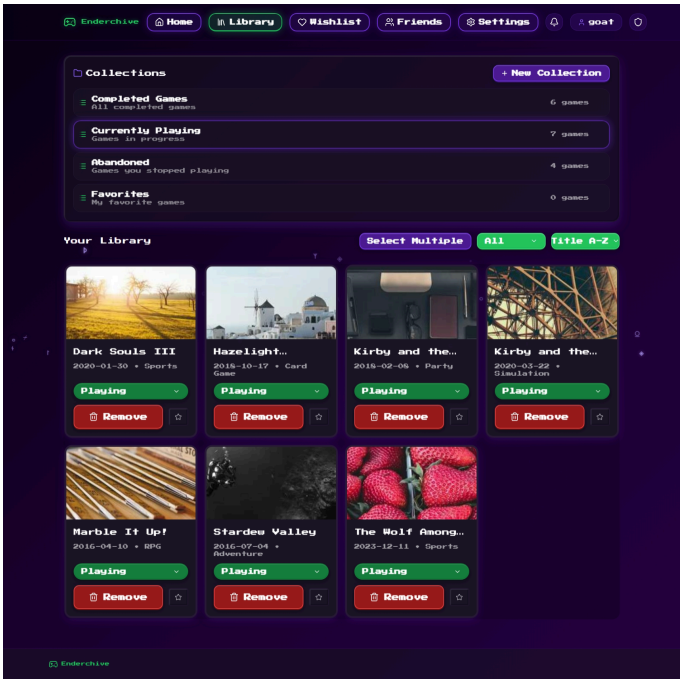
The `UserGameController` handles all library operations. Games are added to the library through the `user_games` table which creates a many-to-many relationship between users and games with additional attributes (status, rating, playtime_hours). The status is stored as an ENUM type for data integrity. The frontend displays games in a responsive grid layout with filtering capabilities. Status updates are handled via API endpoints that validate the status enum value.

Screenshots:









5.3 Feature 3: Game Catalog & Search

Description: Comprehensive game database with search and filtering capabilities to help users discover and find games to add to their library.

Functionality:

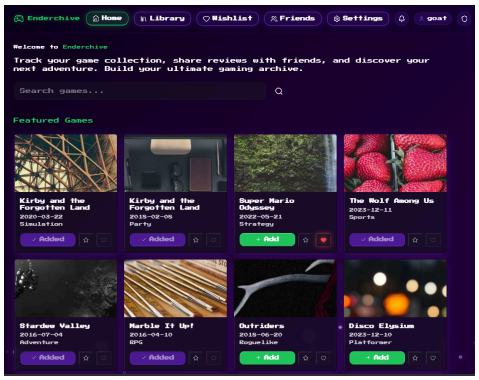
- Browse game catalog
- Search games by title
- Filter games by genre
- Filter games by platform
- Filter games by tags
- View game details (title, description, genre, developer, publisher, release date, platforms, tags, cover image)
- View game reviews and ratings
- Add games to library from catalog

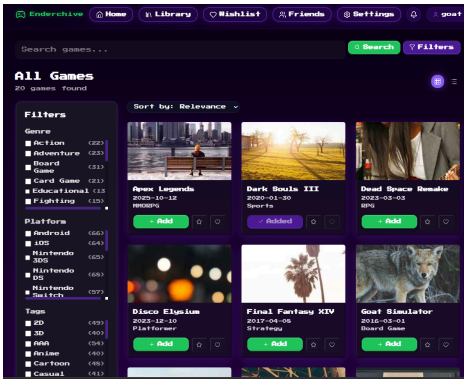
Implementation Details:

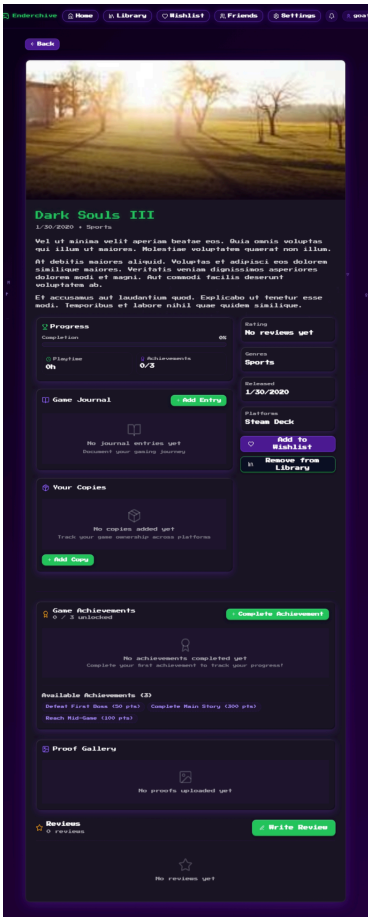
The `GameController` handles game listing and detail views. Games are stored in the `games` table with relationships to genres, platforms (many-to-many), and tags (many-to-many). Search functionality uses Laravel's query builder with LIKE clauses for title matching. Filtering is implemented through whereHas()

methods on relationships. The frontend provides a search input and filter dropdowns that send API requests to fetch filtered results.

Screenshots:







5.4 Feature 4: Game Progress Tracking

Description: System for tracking game progress through milestones and playtime, allowing users to monitor their advancement through games.

Functionality:

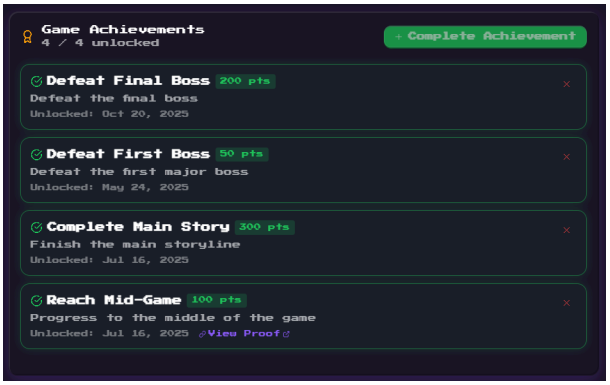
- View available milestones for a game

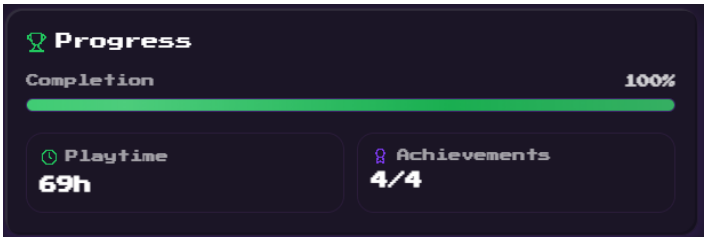
- Mark milestones as completed
- Track completion timestamps
- Update total playtime hours
- View progress percentage (completed milestones vs total milestones)
- See progress history

Implementation Details:

Milestones are stored in the `game_milestones` table, linked to games. User progress is tracked in the `user_game_progress` table, which links users, games, and milestones with a completion timestamp. The `UserGameProgressController` handles creating and deleting progress records. The frontend displays milestones as a list with checkboxes or buttons to mark them complete, and shows progress indicators.

Screenshots:





5.5 Feature 5: Review System

Description: Allow users to write, edit, and delete reviews for games they've played, sharing their opinions with the community.

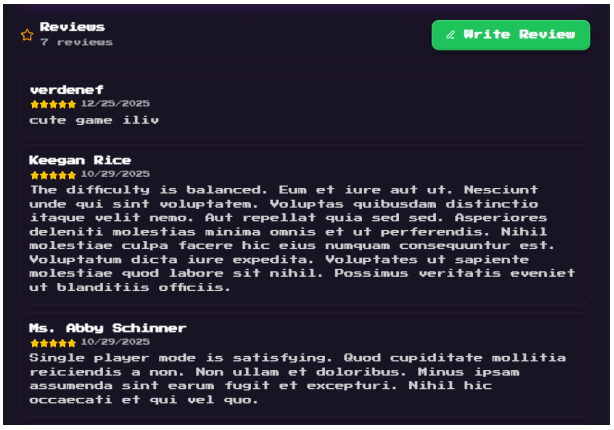
Functionality:

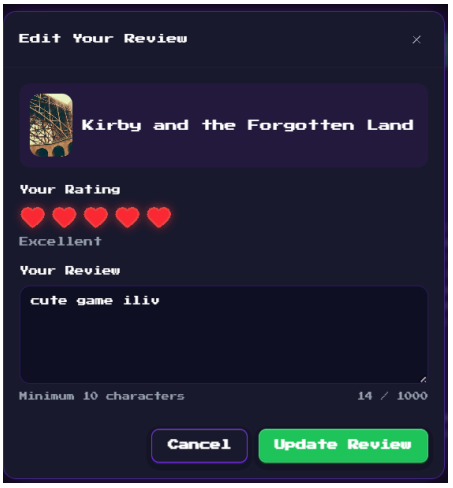
- Create reviews with rating (1-5) and optional text
- Edit own reviews
- Delete own reviews
- View all reviews for a game
- See review author information
- Display average rating for games

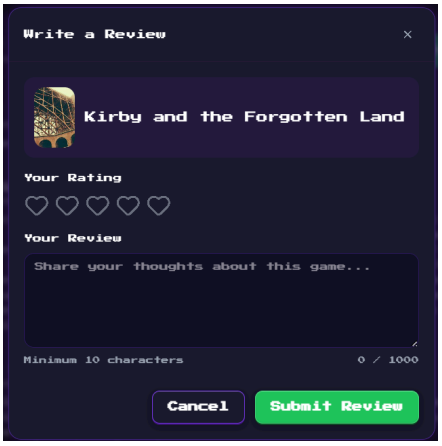
Implementation Details:

Reviews are stored in the `reviews` table with foreign keys to users and games. The `ReviewController` handles CRUD operations with authorization checks to ensure users can only edit/delete their own reviews. Validation ensures ratings are within 1-5 range and review text is optional. The frontend displays reviews in a list format with edit/delete buttons for the review owner.

Screenshots:







5.6 Feature 6: Friend System

Description: Social feature allowing users to connect with friends, send friend requests, and view friends' game libraries and activities.

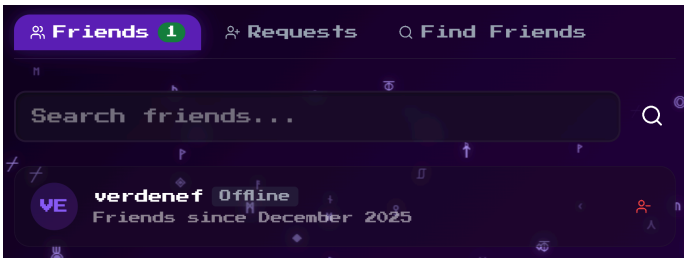
Functionality:

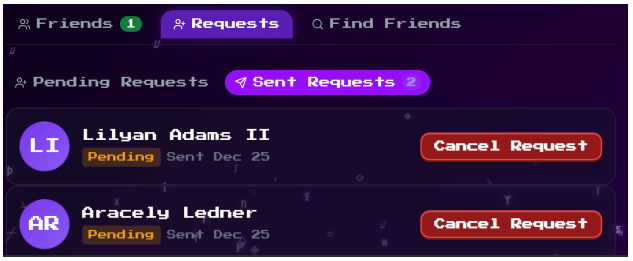
- Search for users by username or name
- Send friend requests
- Accept or decline friend requests
- View pending friend requests
- View friends list
- Remove friends
- View friend profiles and libraries
- See friend activity (reviews, recently added games)

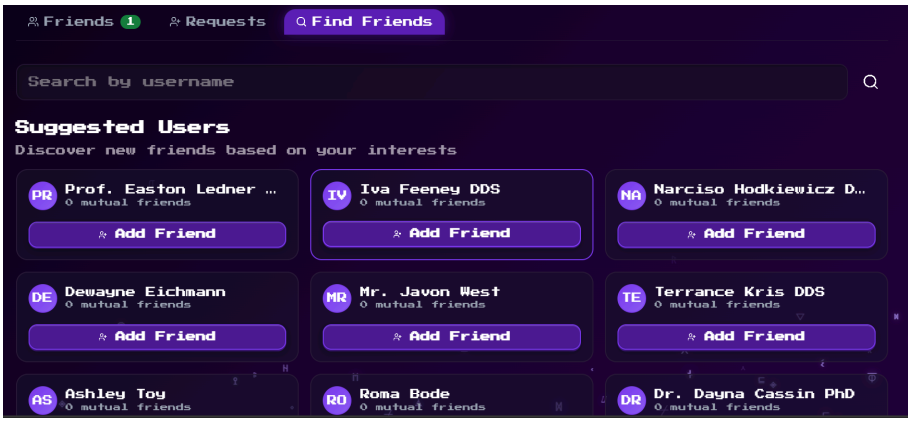
Implementation Details:

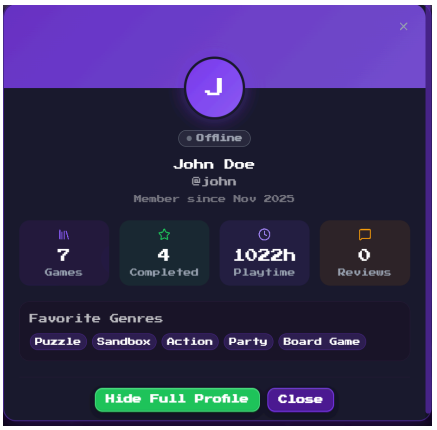
Friend relationships are stored in the `friends` table with status enum (Pending, Accepted, Blocked). The `FriendController` handles all friend operations. The system prevents duplicate friend requests and self-friending. Friend requests are bidirectional in terms of viewing, but stored with user_id (initiator) and friend_user_id (target). The frontend provides separate tabs for friends list, requests, and user search.

Screenshots:









5.7 Feature 7: Game Ownership Management

Description: Track detailed ownership information including game editions, purchase stores, and platforms for comprehensive collection management.

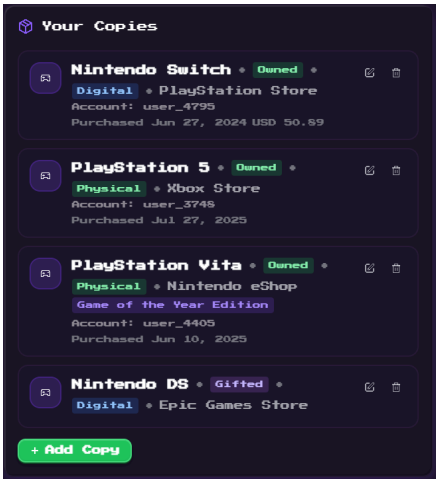
Functionality:

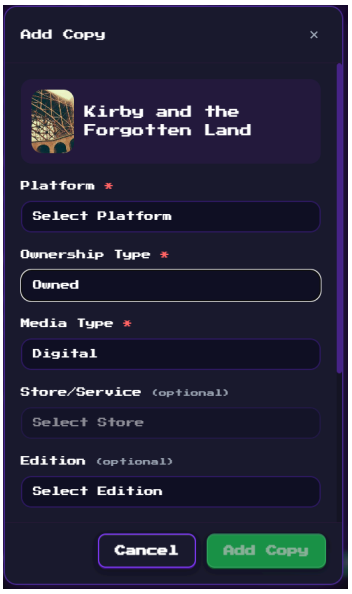
- Record game ownership with edition information
- Specify store where game was purchased
- Specify platform for the ownership
- View all ownership records for a game
- Delete ownership records
- Track multiple ownerships of the same game (different editions/platforms)

Implementation Details:

Ownerships are stored in the `user_game_ownerships` table with foreign keys to users, games, editions, stores, and platforms. The `UserGameOwnershipController` handles CRUD operations. This allows users to track that they own a game in multiple formats (e.g., physical Standard edition for PlayStation, digital Deluxe edition for PC on Steam). The frontend provides forms to add ownership with dropdowns for edition, store, and platform selection.

Screenshots:





5.8 Feature 8: Admin Dashboard

Description: Administrative interface for managing application content including users, games, genres, platforms, and tags.

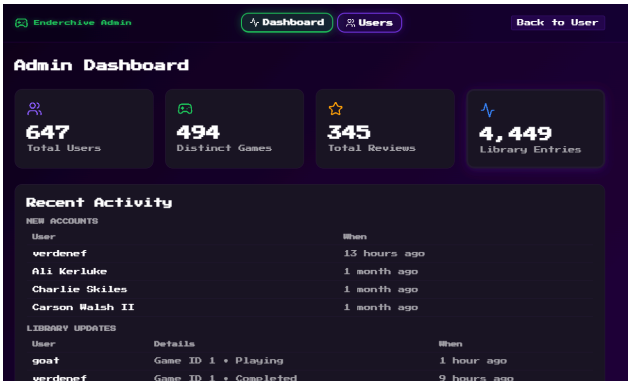
Functionality:

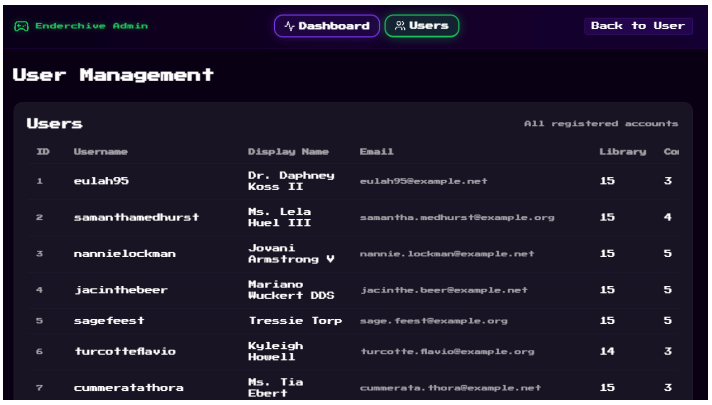
- View dashboard statistics (total users, games, reviews)
- Manage users (view user list, user details)
- Create, update, and delete games
- Create, update, and delete genres
- Create, update, and delete platforms
- Create, update, and delete tags
- View system activity

Implementation Details:

Admin functionality is protected by authorization checks in the `AdminController`. Admin access is determined by checking if the user's email matches the admin email (configurable). The admin dashboard uses the same API endpoints but with additional authorization middleware. Admin routes are protected and return 403 errors for non-admin users. The frontend provides an admin interface with tabs for different management functions.

Screenshots:





---

6. Security & Error Handling

6.1 Security Measures Implemented

- **Authentication:**
- **Authorization:**
- **Input Validation:**
- **Data Protection:**
- **Protection Against Vulnerabilities:**

6.2 Error Handling

The application implements comprehensive error handling at multiple levels:

Backend Error Handling:

1. **Validation Errors:**
2. **Authentication Errors:**
3. **Authorization Errors:**
4. **Not Found Errors:**

5. **Server Errors:**

Frontend Error Handling:

1. **API Error Responses:**
2. **Form Validation Errors:**
3. **Network Errors:**
4. **User Feedback:**

Error Handling Examples:

- **Registration Validation:** If username is already taken, user sees: "A user with this username already exists"
- **Login Failure:** If credentials are incorrect, user sees: "Unauthorized" message
- **Game Not Found:** If accessing non-existent game, API returns 404 with message: "Resource not found"
- **Unauthorized Action:** If user tries to delete another user's review, API returns 403: "Unauthorized access"

---

7. Installation & Setup Instructions

7.1 Prerequisites

- PHP 8.2 or higher
- Composer (PHP dependency manager)
- Node.js 18+ and NPM (Node Package Manager)
- MySQL 8.0+ (Primary database)
- Git (for cloning repository)

7.2 Installation Steps

1. **Clone the repository:**git clone [your-repo-url]
cd enderchive-laravel
2. **Install PHP dependencies:**composer install
3. **Install Node.js dependencies:**npm install
4. **Configure environment variables:**
5. **Set up the database:**
6. **Build frontend assets:**npm run buildOr for development with hot reload:npm run dev
7. **Run the application:**

Alternative: Using the provided batch file (Windows):

1. Run `run-enderchive.bat` from the project root directory
2. This will automatically start both Laravel and Vite servers

7.3 Running the Application

Once the application is running:

- **Access URL:** `http://localhost:8000`
- **Default Admin Account:**
- **Initial Setup:**
- **API Endpoints:** Available at `http://localhost:8000/api/*`
- **Development Mode:**

Note: For production deployment, ensure to:

- Set `APP_ENV=production` in `.env`
- Set `APP_DEBUG=false` in `.env`
- Run `npm run build` to compile production assets
- Configure proper database credentials
- Set up proper web server (Apache/Nginx) configuration

---

8. Testing

8.1 Test Cases

Test Case	**Steps**	**Expected Result**	**Actual Result**	**Status**
User Registration	1. Navigate to Sign Up page 2. Fill in all required fields 3. Submit form	User account created, redirected to home page, logged in automatically	User account created successfully	Pass
User Login	1. Navigate to Login page 2. Enter email/username and password 3. Click Login	User authenticated, redirected to home page, session created	Login successful	Pass

Test Case	**Steps**	**Expected Result**	**Actual Result**	**Status**
Add Game to Library	1. Search for a game 2. Click on game 3. Click "Add to Library" 4. Select status and submit	Game added to user's library with selected status	Game appears in library	Pass
Update Game Status	1. Go to Library page 2. Click on a game 3. Change status 4. Save changes	Game status updated in database and UI	Status updated successfully	Pass
Write Review	1. Navigate to game detail page 2. Click "Write Review" 3. Enter rating and text 4. Submit	Review created and displayed on game page	Review saved and visible	Pass
Edit Review	1. Go to game detail page 2. Find own review 3. Click Edit 4. Modify content 5. Save	Review updated in database and UI	Review updated successfully	Pass
Delete Review	1. Go to game detail page 2. Find own review 3. Click Delete 4. Confirm deletion	Review removed from database and UI	Review deleted successfully	Pass
Send Friend Request	1. Go to Friends page 2. Search for user 3. Click "Send Friend Request"	Friend request created with "Pending" status	Request sent successfully	Pass
Accept Friend Request	1. Go to Friends page2. Navigate to Requests tab3. Click Accept on pending request	Friend status changed to "Accepted"	Friend request accepted	Pass
Search Games	1. Go to Home page2. Enter game title in search3. Press Enter or click search	Matching games displayed in results	Search results shown correctly	Pass
Filter Games by Genre	1. Go to Search page 2. Select genre from filter 3. View results	Only games with selected genre displayed	Filter working correctly	Pass
Track Milestone Progress	1. Go to game detail page2. Navigate to Milestones section 3. Click "Mark Complete" on milestone	Milestone marked as completed, progress updated	Progress tracked successfully	Pass
Update Profile	1. Go to Settings page 2. Modify name, username, or bio	Profile information updated in database	Profile updated successfully	Pass

Test Case	**Steps**	**Expected Result**	**Actual Result**	**Status**
	3. Save changes			
Change Password	1. Go to Settings page 2. Enter current and new password 3. Submit	Password updated, user can login with new password	Password changed successfully	Pass
Admin Dashboard Access	1. Login as admin user 2. Navigate to /admin	Admin dashboard displayed with statistics	Dashboard accessible	Pass
Non-Admin Access Denied	1. Login as regular user 2. Attempt to access /admin	403 Forbidden error or redirect	Access properly restricted	Pass
Game CRUD (Admin)	1. Login as admin 2. Create new game 3. Update game details 4. Delete game	Game created, updated, and deleted successfully	CRUD operations working	Pass
Validation - Duplicate Email	1. Attempt to register with existing email	Validation error displayed, registration prevented	Validation working	Pass
Validation - Weak Password	1. Attempt to register with password < 8 characters	Validation error displayed	Password validation working	Pass
Unauthorized Review Edit	1. Login as User A 2. Attempt to edit User B's review	403 Forbidden error	Authorization working	Pass

8.2 Known Issues & Limitations

Known Issues:

1. **Image Upload:** Currently, game cover images and user avatars are stored as URLs. File upload functionality for images is not fully implemented. Users must provide external image URLs.

2. **Real-time Notifications:** Friend request notifications and activity updates are not pushed in real-time. Users need to refresh the page to see new friend requests.

3. **Password Reset:** Password reset functionality via email is not implemented. Users cannot recover forgotten passwords through the application.

4. **Search Performance:** Search functionality may be slow with large datasets. Full-text search indexing is not implemented.

5. **Mobile Responsiveness:** Some pages may have minor layout issues on very small mobile screens (< 320px width).

Limitations:

1. **No Email Functionality:** The application does not send emails for registration confirmations, password resets, or notifications.

2. **No External API Integration:** Game data must be entered manually. No integration with external game databases (IGDB, RAWG, etc.).

3. **No Social Media Integration:** Users cannot share their game libraries or reviews on social media platforms.

4. **No Advanced Statistics:** Limited statistics and analytics features. No charts or graphs for gaming trends.

5. **No Export Functionality:** Users cannot export their game library data to CSV, JSON, or other formats.

Future Improvements:

1. Implement file upload for game covers and avatars
2. Add email notification system
3. Integrate with external game APIs for automatic game data
4. Implement real-time notifications using WebSockets
5. Add advanced search with full-text indexing
6. Create mobile native applications
7. Add data export functionality
8. Implement password reset via email
9. Add more comprehensive statistics and analytics
10. Improve mobile responsiveness for all screen sizes

---

9. Code Quality & Documentation

9.1 Code Structure

The project follows Laravel's standard directory structure with clear separation of concerns:

```
enderchive-laravel/
├── app/
│   ├── Http/
│   │   ├── Controllers/
│   │   │   ├── API/          # API Controllers (RESTful)
│   │   │   └── Auth/         # Authentication Controllers
│   │   ├── Middleware/       # Custom middleware
│   │   └── Requests/         # Form Request validation classes
│   └── Models/               # Eloquent models
├── bootstrap/                # Application bootstrap files
├── config/                   # Configuration files
├── database/
│   ├── factories/            # Model factories for testing
│   ├── migrations/           # Database migrations
│   └── seeders/              # Database seeders
├── public/                   # Public assets and entry point
```

```
├── resources/
│   ├── css/                    # CSS files
│   ├── js/
│   │   ├── Components/         # Vue.js reusable components
│   │   ├── Pages/              # Vue.js page components
│   │   ├── Layouts/            # Layout components
│   │   └── Composables/        # Vue composables
│   └── views/                  # Blade templates (minimal, using Inertia)
├── routes/
│   ├── api.php                 # API routes
│   └── web.php                 # Web routes
├── storage/                    # Logs, cache, uploaded files
├── tests/                       # PHPUnit tests
└── vendor/                     # Composer dependencies
```

9.2 Code Standards

Naming Conventions:

- **Classes:** PascalCase (e.g., `GameController`, `UserGame`)
- **Methods/Functions:** camelCase (e.g., `getUserGames()`, `updateProfile()`)
- **Variables:** camelCase (e.g., `userGames`, `gameStatus`)
- **Constants:** UPPER_SNAKE_CASE (e.g., `MAX_RATING`)
- **Database Tables:** snake_case, plural (e.g., `user_games`, `game_milestones`)
- **Database Columns:** snake_case (e.g., `user_id`, `created_at`)
- **Vue Components:** PascalCase (e.g., `GameItemCard.vue`, `NavigationTab.vue`)

Code Style:

- PHP code follows PSR-12 coding standards
- Laravel Pint configured for automatic code formatting
- JavaScript/Vue code uses ESLint (if configured)
- Consistent indentation (4 spaces for PHP, 2 spaces for JavaScript)

Comments & Documentation:

- PHPDoc comments for all classes and methods
- Inline comments for complex logic
- README.md with setup instructions
- API endpoint documentation in code comments
- Model relationships documented in model classes

Version Control:

- Git for version control
- Meaningful commit messages
- Feature-based branching strategy (if used)

- `.gitignore` properly configured to exclude vendor, node_modules, etc.

9.3 API Endpoints

Authentication Endpoints:

POST /api/register

- Description: Register a new user account
- Request: `{ "name": "string", "username": "string", "email": "string", "password": "string", "password_confirmation": "string" }`
- Response: `{ "response_code": 201, "status": "success", "message": "Successfully registered", "data": { "user": {...}, "token": "string" } }`
- Status Codes: 201 (Created), 422 (Validation Error), 500 (Server Error)

POST /api/login

- Description: Authenticate user and receive API token
- Request: `{ "email": "string", "password": "string" }`
- Response: `{ "response_code": 200, "status": "success", "data": { "user": {...}, "token": "string" } }`
- Status Codes: 200 (Success), 401 (Unauthorized), 422 (Validation Error)

POST /api/logout (Protected)

- Description: Logout user and revoke API token
- Request: Bearer token in Authorization header
- Response: `{ "response_code": 200, "status": "success", "message": "Logged out successfully" }`
- Status Codes: 200 (Success), 401 (Unauthorized)

GET /api/get-user (Protected)

- Description: Get authenticated user information
- Request: Bearer token in Authorization header
- Response: `{ "response_code": 200, "status": "success", "data": { "user": {...} } }`
- Status Codes: 200 (Success), 401 (Unauthorized)

Game Endpoints:

GET /api/games

- Description: Get list of games (public)

- Query Parameters: `search` (string, optional), `genre_id` (int, optional), `platform_id` (int, optional)
- Response: `{ "data": [{ "game_id": int, "title": "string", ... }] }`
- Status Codes: 200 (Success)

GET /api/games/{game}

- Description: Get specific game details (public)
- Response: `{ "data": { "game_id": int, "title": "string", "description": "string", ... } }`
- Status Codes: 200 (Success), 404 (Not Found)

POST /api/games (Protected)

- Description: Create a new game
- Request: `{ "title": "string", "description": "string", "genre_id": int, "platform_ids": [int], ... }`
- Response: `{ "response_code": 201, "status": "success", "data": { "game": {...} } }`
- Status Codes: 201 (Created), 422 (Validation Error), 401 (Unauthorized)

PUT /api/games/{game} (Protected)

- Description: Update a game
- Request: Same as POST
- Response: `{ "response_code": 200, "status": "success", "data": { "game": {...} } }`
- Status Codes: 200 (Success), 404 (Not Found), 422 (Validation Error), 401 (Unauthorized)

DELETE /api/games/{game} (Protected)

- Description: Delete a game
- Response: `{ "response_code": 200, "status": "success", "message": "Game deleted successfully" }`
- Status Codes: 200 (Success), 404 (Not Found), 401 (Unauthorized)

User Game (Library) Endpoints:

GET /api/user/games (Protected)

- Description: Get user's game library
- Query Parameters: `status` (enum, optional: Wishlist, Playing, Completed, Abandoned)
- Response: `{ "data": [{ "user_game_id": int, "game": {...}, "status": "string", ... }] }`
- Status Codes: 200 (Success), 401 (Unauthorized)

POST /api/user/games (Protected)

- Description: Add game to user's library
- Request: `{ "game_id": int, "status": "string", "rating": int, "playtime_hours": int }`
- Response: `{ "response_code": 201, "status": "success", "data": { "user_game": {...} } }`
- Status Codes: 201 (Created), 422 (Validation Error), 401 (Unauthorized)

PUT /api/user/games/{userGame} (Protected)

- Description: Update user's game entry
- Request: Same as POST
- Response: `{ "response_code": 200, "status": "success", "data": { "user_game": {...} } }`
- Status Codes: 200 (Success), 404 (Not Found), 422 (Validation Error), 401 (Unauthorized)

DELETE /api/user/games/{userGame} (Protected)

- Description: Remove game from user's library
- Response: `{ "response_code": 200, "status": "success", "message": "Game removed from library" }`
- Status Codes: 200 (Success), 404 (Not Found), 401 (Unauthorized)

Review Endpoints:

GET /api/games/{game}/reviews (Public)

- Description: Get all reviews for a game
- Response: `{ "data": [{ "review_id": int, "user": {...}, "rating": int, "review_text": "string", ... }] }`
- Status Codes: 200 (Success), 404 (Not Found)

POST /api/games/{game}/reviews (Protected)

- Description: Create a review for a game
- Request: `{ "rating": int, "review_text": "string" }`
- Response: `{ "response_code": 201, "status": "success", "data": { "review": {...} } }`
- Status Codes: 201 (Created), 422 (Validation Error), 401 (Unauthorized)

PUT /api/reviews/{review} (Protected)

- Description: Update own review
- Request: Same as POST
- Response: `{ "response_code": 200, "status": "success", "data": { "review": {...} } }`
- Status Codes: 200 (Success), 403 (Forbidden), 404 (Not Found), 401 (Unauthorized)

DELETE /api/reviews/{review} (Protected)

- Description: Delete own review
- Response: `{ "response_code": 200, "status": "success", "message": "Review deleted successfully" }`
- Status Codes: 200 (Success), 403 (Forbidden), 404 (Not Found), 401 (Unauthorized)

Friend Endpoints:

GET /api/friends (Protected)

- Description: Get user's friends list
- Response: `{ "data": [{ "friend_id": int, "friend_user": {...}, "status": "string", ... }] }`
- Status Codes: 200 (Success), 401 (Unauthorized)

GET /api/friends/requests (Protected)

- Description: Get pending friend requests
- Response: `{ "data": [{ "friend_id": int, "user": {...}, "status": "Pending", ... }] }`
- Status Codes: 200 (Success), 401 (Unauthorized)

GET /api/friends/search (Protected)

- Description: Search for users to add as friends
- Query Parameters: `q` (string, required)
- Response: `{ "data": [{ "user_id": int, "name": "string", "username": "string", ... }] }`
- Status Codes: 200 (Success), 401 (Unauthorized)

POST /api/friends (Protected)

- Description: Send friend request
- Request: `{ "friend_user_id": int }`
- Response: `{ "response_code": 201, "status": "success", "data": { "friend": {...} } }`
- Status Codes: 201 (Created), 422 (Validation Error), 401 (Unauthorized)

PUT /api/friends/{friend} (Protected)

- Description: Accept or update friend request
- Request: `{ "status": "Accepted" }`
- Response: `{ "response_code": 200, "status": "success", "data": { "friend": {...} } }`
- Status Codes: 200 (Success), 404 (Not Found), 422 (Validation Error), 401 (Unauthorized)

DELETE /api/friends/{friend} (Protected)

- Description: Remove friend or decline request
- Response: `{ "response_code": 200, "status": "success", "message": "Friend removed successfully" }`
- Status Codes: 200 (Success), 404 (Not Found), 401 (Unauthorized)

Admin Endpoints:

GET /api/admin/summary (Protected - Admin Only)

- Description: Get admin dashboard statistics
- Response: `{ "data": { "total_users": int, "total_user_games": int, "distinct_games": int, "total_reviews": int } }`
- Status Codes: 200 (Success), 403 (Forbidden), 401 (Unauthorized)

GET /api/admin/users (Protected - Admin Only)

- Description: Get list of all users with statistics
- Response: `{ "data": [{ "user_id": int, "name": "string", "email": "string", "stats": {...}, ... }] }`
- Status Codes: 200 (Success), 403 (Forbidden), 401 (Unauthorized)

Other Endpoints:

- **GET /api/platforms** - Get all platforms (Public)
- **GET /api/genres** - Get all genres (Public)
- **GET /api/tags** - Get all tags (Public)
- **GET /api/stores** - Get all stores (Public)
- **GET /api/games/{game}/milestones** - Get milestones for a game (Public)
- **GET /api/games/{game}/editions** - Get editions for a game (Public)
- **PUT /api/user/profile** - Update user profile (Protected)
- **PUT /api/user/password** - Change password (Protected)
- **PUT /api/user/username** - Update username (Protected)
- **GET /api/users/{userId}/profile** - Get user profile (Protected)
- **GET /api/user/games/{game}/ownerships** - Get ownerships for a game (Protected)
- **POST /api/user/games/{game}/ownerships** - Add ownership (Protected)
- **DELETE /api/user/games/{game}/ownerships/{ownership}** - Remove ownership (Protected)
- **POST /api/user/games/{game}/progress/{milestone}** - Mark milestone complete (Protected)
- **DELETE /api/user/games/{game}/progress/{milestone}** - Unmark milestone (Protected)

---

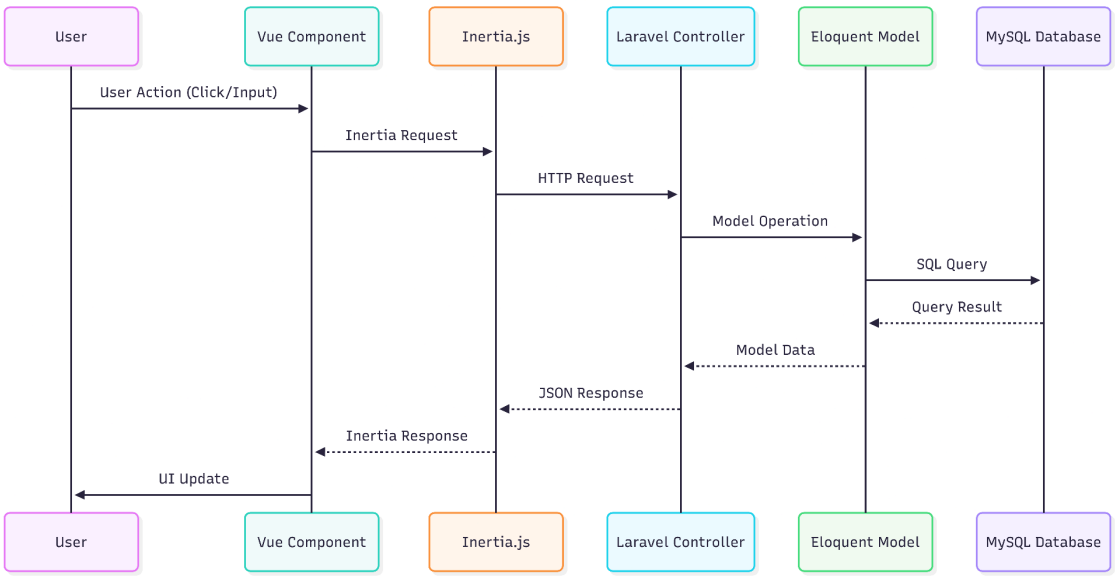
10. References & Resources

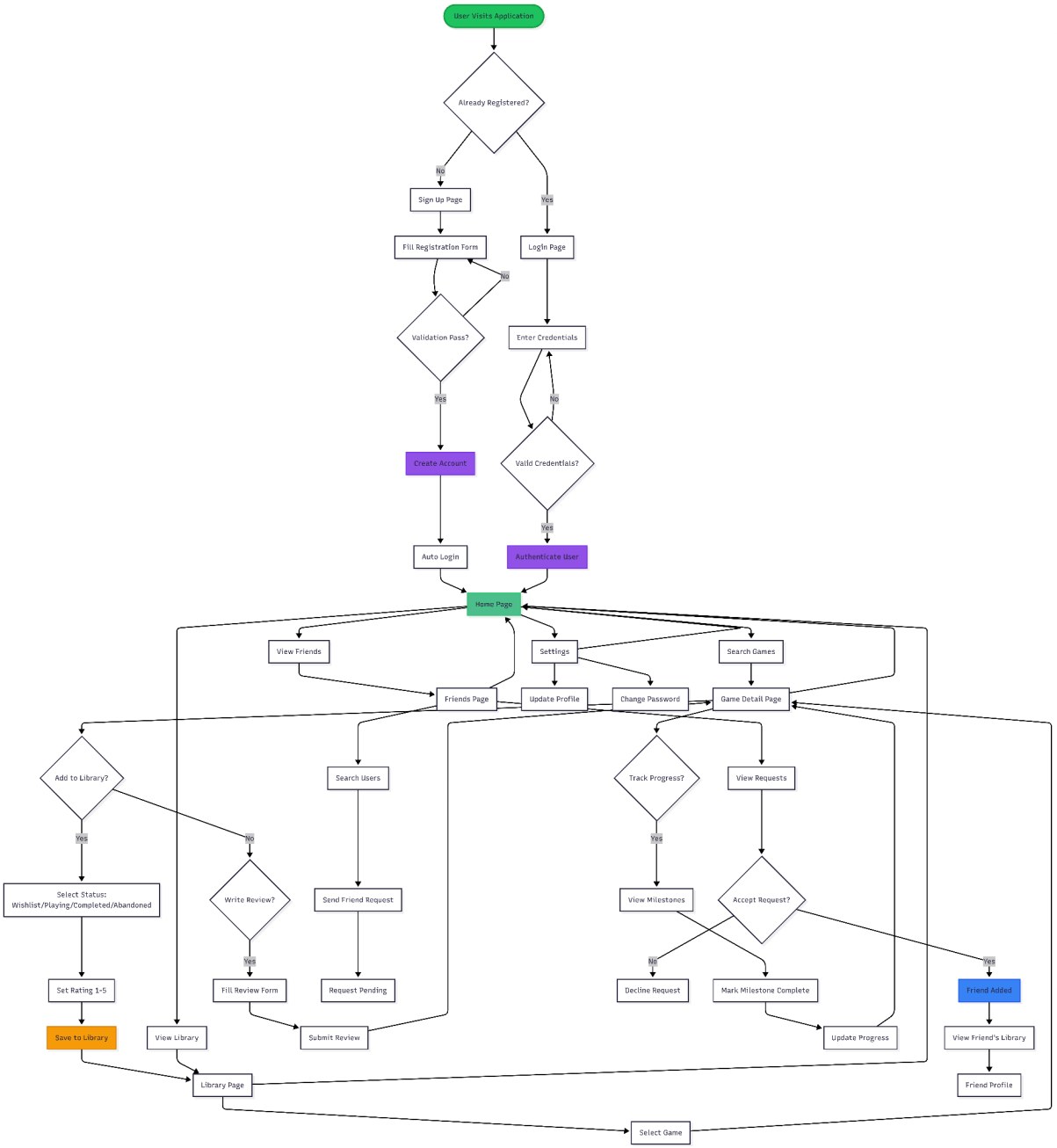
- **Laravel Documentation:** https://laravel.com/docs
- **Vue.js Documentation:** https://vuejs.org/
- **Inertia.js Documentation:** https://inertiajs.com/
- **Tailwind CSS Documentation:** https://tailwindcss.com/docs
- **Laravel Sanctum Documentation:** https://laravel.com/docs/sanctum
- **Eloquent ORM Documentation:** https://laravel.com/docs/eloquent
- **Vite Documentation:** https://vitejs.dev/
- **PHP Documentation:** https://www.php.net/docs.php
- **MySQL Documentation:** https://dev.mysql.com/doc/
- **Git Documentation:** https://git-scm.com/doc
- **Composer Documentation:** https://getcomposer.org/doc/
- **NPM Documentation:** https://docs.npmjs.com/
- **MDN Web Docs (JavaScript):** https://developer.mozilla.org/en-US/docs/Web/JavaScript
- **W3Schools (HTML/CSS):** https://www.w3schools.com/

---

11. Appendix

11.1 Additional Diagrams





11.2 Supplementary Information

Environment Variables (.env):

Key environment variables used in the application:

- `APP_NAME` - Application name
- `APP_ENV` - Environment (local, production)
- `APP_KEY` - Application encryption key

- `APP_DEBUG` - Debug mode (true/false)
- `APP_URL` - Application URL
- `DB_CONNECTION` - Database driver (mysql for MySQL, pgsql for PostgreSQL)
- `DB_DATABASE` - Database name or path
- `DB_HOST` - Database host (for MySQL/PostgreSQL)
- `DB_PORT` - Database port
- `DB_USERNAME` - Database username
- `DB_PASSWORD` - Database password

Database Seeding:

The application includes database seeders for:

- Users (including admin user)
- Games
- Genres
- Platforms
- Tags
- Stores
- Sample relationships (user_games, reviews, friends)

Run seeders with: `php artisan db:seed`

Performance Considerations:

- Database queries optimized with eager loading (`with()` method)
- Indexes on foreign keys for faster joins
- Frontend assets compiled and minified for production
- Vite HMR for fast development iteration
- API responses paginated where appropriate (can be implemented)

Security Best Practices Implemented:

- All passwords hashed using bcrypt
- CSRF protection on all forms
- SQL injection prevention via Eloquent ORM
- XSS prevention via automatic escaping
- Input validation on all endpoints
- Authorization checks on protected resources
- Secure session management

- API token authentication

Browser Compatibility:

- Modern browsers (Chrome, Firefox, Safari, Edge)
- Responsive design tested on various screen sizes
- Mobile browsers supported
- Progressive enhancement approach

Student Signature: Van Renfred M. Otacan