FPT UNIVERSITY

Capstone Project Document

FPT GameHub Platform

Group Code: SP25SE002	
Group Members	Quách Hoàng Huy - Team Leader - SE172520 Vũ Lê Lâm Hoàng - Team Member - SE172277 Nguyễn Hoàng Anh - Team Member - SE172090 Trần Thẩm Anh Toàn - Team Member - SE171871 Bạch Doãn Vương - Team Member - DE160256
Supervisor	Phạm Thanh Trí
Ext Supervisor	
Capstone Project code	GSP25SE02

- Ho Chi Minh, April 2025 -

Table of Contents

List of Tables

List of Figures

Acknowledgement

We would like to express our sincere gratitude to our lecturer, Mr. Phạm Thanh Trí, for his continuous guidance, valuable feedback, and encouragement throughout the development of this project.

We also extend our appreciation to all team members of GameHub Platform Team for their dedication, collaboration, and collective efforts in bringing the GameHub Platform for FPTU Campus to life. Each member contributed significantly in different aspects of the project, and this achievement is the result of our strong teamwork and shared vision.

Lastly, we acknowledge the support and inspiration from our peers and the FPT University environment, which provided us the opportunity and resources to explore our skills and creativity in software development.

Thank you for your support.

Best regards,

GameHub Platform Team

Definition and Acronyms

Acronym	Definition

PWM	Psychology website
AWS	Amazon Web Services
BA	Business Analysis
BR	Business Rule
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
PM	Project Manager
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirement Specification
UAT	User Acceptance Test
UC	Use Case
API	Application Program Interface
FE	Features
LI	Limitation
WBS	Work Breakdown Structure
UI	User Interface
CI/CD	Continuous Integration and Continuous Delivery
FE	Frontend
BE	Backend

Table 1 - Acronym and Definition

I. Project Introduction

1. Overview

1.1 Project Information

- Project name: Game Hub platform for games and design assets for FPTU Campus students
- Project code: GSP25SE02
- Group name: SP25SE002
- Software type: Web Application, Online Services

1.2 Project Team

Full Name	Role	Email	Mobile
Phạm Thanh Trí	Lecturer	tript9@fpt.edu.vn	
Quách Hoàng Huy	Leader	huyqhse172520@fpt.edu.vn	
Vũ Lê Lâm Hoàng	Member	hoangvllse172277@fpt.edu.vn	
Nguyễn Hoàng Anh	Member	anhnhse172090@fpt.edu.vn	
Trần Thẩm Anh Toàn	Member	toanttase171871@fpt.edu.vn	
Bạch Doãn Vương	Member	vuongbdde160256@fpt.edu.vn	

Table 2 - Project Team

2. Product Background

The gaming industry has witnessed unprecedented growth, attracting a vast audience of enthusiasts and developers. With a thriving community dedicated to game creation, there is a significant demand for a platform that supports the building, development, and publishing of games. This need is particularly pronounced among students and young developers, FPTU students who are interested in building games as well as students following the course PRU221, these people often face financial constraints but have a strong desire to share and distribute their creations.

Game hub platform, a web application - provide a platform and online services aims to develop a comprehensive game hub platform that caters to the above demographic. This platform will offer robust tools and resources to facilitate game development and provide an accessible avenue for publishing and sharing games with a broader audience. By addressing the specific needs of aspiring game developers, our game hub will empower a new generation of creators to bring their innovative ideas to life and connect with a global community of gamers and developers.

3. Existing Systems

3.1 Steam

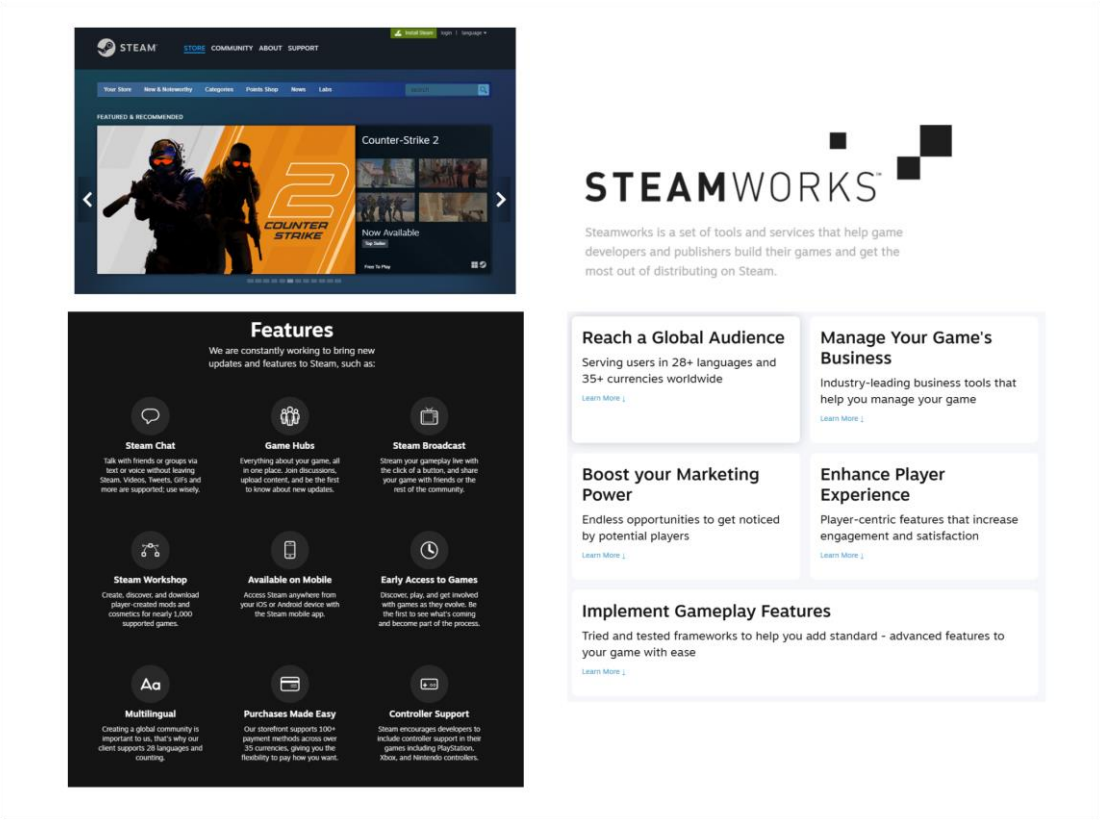

Figure 1 - Steam Features Overview

Steam is a digital distribution service and storefront developed by Valve Corporation. It is the ultimate destination for playing, discussing, and creating games.

Link web:

- Steam for Player: https://store.steampowered.com/about/
- Steam for developer: https://partner.steamgames.co

3.2 Epic Game

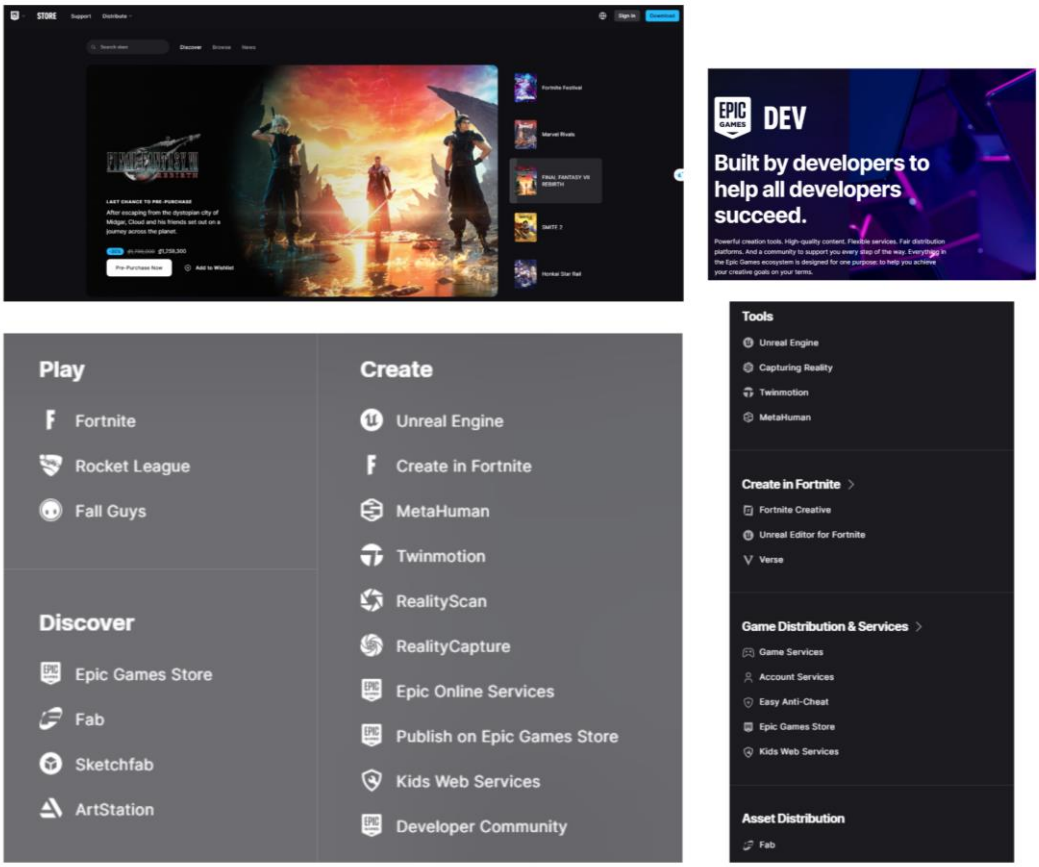

Figure 2 - Steam Features Overview

Epic Games is an American company founded by CEO Tim Sweeney, provides an end-to-end digital ecosystem for developers and creators to build, distribute, and operate games and other content.

Link web:

- Epic Game for Player**:** https://www.epicgames.com/site/en-US/about
- Epic Game for Developer: https://dev.epicgames.com/en-US

4. Business Opportunity

The proposed web platform addresses the growing demand for developers and designers to monetize their creative works in the gaming industry. This platform allows developers to upload and sell their games, while designers can upload and sell game assets such as 3D models, animations, and textures. Users can purchase these games and assets directly, and also download games for play.

The current market lacks a unified and transparent platform tailored specifically for indie developers and designers to showcase and sell their products, providing them with visibility and revenue opportunities. Existing solutions often either lack specialization in game-related content or impose high fees and restrictions, limiting accessibility for smaller creators.

Our platform aligns with the trend of democratizing game development by empowering creators with tools to sell and promote their work while enabling users to access high-quality games and assets conveniently. It bridges a gap in the market by fostering collaboration between developers and designers, enhancing productivity and creativity in the gaming ecosystem.

5. Software Product Vision

For students and young developers, especially FPTU students and those taking the PRU221 course, who want to create, develop, and publish games, GameHub Platform is a web application that provides easy-to-use tools and resources for developing, publishing, and trading games and assets. GameHub empowers and offers opportunities to students passionate about game creation and aspiring game developers facing financial challenges. It provides a solution to turn their creative ideas into reality, earn profits, gain experience in game development, and connect with communities of gamers and game designers, including players and developers.

6. Project Scope & Limitations

6.1 Major Features

❖ Customer Role

Account

- **FE-01. Account Management:** Register, login, view and update account profile, reset password if forgotten, and logout.

Game & Assets

- **FE-02. Game Products:** Search, filter, view detailed information, and manage game products and related assets.
- **FE-03. Cart/Wishlist:** Add or remove game products from the cart or wishlist for later purchase.
- **FE-04. Purchase History:** View a history of purchased game products, and manage payment details.
- **FE-05. Purchasing Process:** Complete purchases by adding items to the cart and proceeding to checkout using various payment methods.
- **FE-06.** Download Purchased Game Asset And Game Products: download purchased game assets and game products from the library.

Support

- **FE-07. Support Tickets:** Submit, view, and track the status of support tickets for issues or inquiries related to purchases or game products.

Notification

- **FE-08. Notification Management:** View notifications about game releases, promotions, or updates related to support tickets, and manage preferences for notifications.

❖ Developer Role

Account

- **FE-01. Account Management**: Register, login, view and update account profile, reset password if forgotten, and logout.

Game Management

- **FE-02. Game Products & Builds**: Create, update, and remove game products and manage associated builds (e.g., game version, patches, updates).
- **FE-03. Upload Builds**: Upload new game builds, patches, or updates for game products.
- **FE-04. Game Release**: Trigger the release of game products once all required criteria are met, and manage release timelines.

Player Account Management

- **FE-05. Player Account Integration**: Access, view, and update player account details through the platform's API.

Game Item Management

- **FE-06. Manage Game Items**: Create, update, and remove game items such as in-game assets, skins, and other collectibles.

API Integration

- **FE-07. API Integration for Game Data**: Integrate GameHub's APIs for managing game player data and items within the platform.

Analytics

- **FE-08. Game Product Analytics**: Track usage, downloads, and performance metrics of the game products released.

Game Release Management

- **FE-09. Release Assets:** Release game products for public or internal use, either immediately or based on a scheduled release.
- **FE-10. Release Requests:** Send requests for game products releases, ensuring they meet the specified criteria for approval.

❖ Designer Role

Account

- **FE-01. Account Management:** Register, login, view and update account profile, reset password if forgotten, and logout.

Asset Management

- **FE-02. Manage Game Assets:** Create, update, view, and remove game assets for specific projects.
- **FE-03. Upload Assets:** Upload game assets, including models, textures, animations, sounds, and other resources necessary for game development.

Game Release Management

- **FE-04. Release Assets:** Release game assets for public or internal use, either immediately or based on a scheduled release.
- **FE-05. Release Requests:** Send requests for game asset releases, ensuring they meet the specified criteria for approval.

❖ Admin Role

Account

- **FE-01. Account Management:** Register, login, view and update account profile, reset password if forgotten, and logout.
- **FE-02. Manage User Accounts:** Create, update, deactivate, or delete user accounts for Customers, Developers, and Designers.
- **FE-03. Role and Permission Management:** Assign roles (Admin, Moderator, Developer, Designer, Customer) and manage permissions within the platform.

Analytics & Reporting

- **FE-04. View Analytics Dashboard:** Access and monitor platform-wide usage, sales, and user activity data.
- **FE-05. Generate Reports:** Generate various reports for financial tracking, user activity, game product sales, and system performance.

System & Configuration

- **FE-06. Platform Configuration:** Manage and configure system-wide settings such as payment gateway settings, security options, and system integrations.

❖ Moderator Role

Account

- **FE-01. Account Management**: View and update moderator profile, change password, and manage login information.

Content Moderation

- **FE-02. Approve/Reject Game Products & Packages**: Review and approve or reject game products and packages, including their content and releases.
- **FE-03. Enforce Community Guidelines**: Flag inappropriate content and enforce community rules by issuing warnings or bans to users who violate guidelines.

Support

- **FE-04. Manage Support Tickets**: Review, manage, and respond to support tickets raised by users for prompt issue resolution.

- **FE-05. Handle User Complaints**: Investigate user complaints, address issues related to other users or platform functionality.

Discount Promotions

- **FE-06. Create Platform Discount Promotions**: Manage and create promotional campaigns related to discounts on game products or packages.

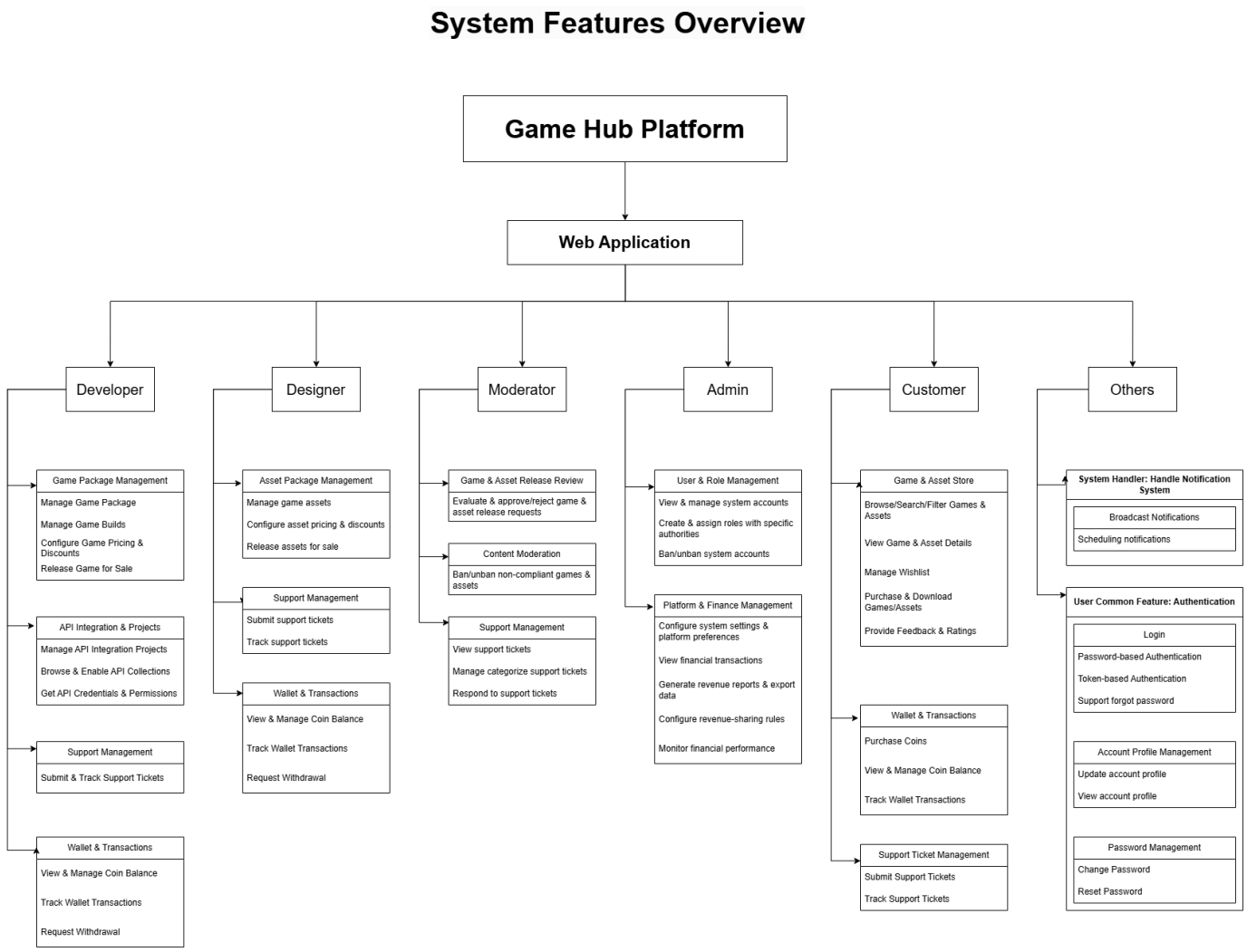

Figure 3 - GameHub Features Overview

6.2 Limitations & Exclusions

LI-1: The platform will not provide in-depth game development tools such as game engines (e.g., Unity, Unreal Engine) or integrated development environments (IDEs). Users are expected to use external tools for development.

LI-2: GameHub will not host games requiring high server performance for real-time multiplayer functionalities. Developers must arrange external server hosting for such games.

LI-3: The platform will not support cryptocurrency-based payments or blockchain technology for transactions during the initial implementation phase.

LI-4: No offline functionality will be provided; the platform requires an active internet connection to access features such as purchases, downloads, and community engagement.

LI-5: GameHub will not offer native mobile or desktop applications initially. The platform will only be available as a web application optimized for desktop and mobile browsers.

LI-6: Moderation tools will not include AI-powered content review or automated compliance checks during the initial release phase. Reviews will rely on manual evaluation by moderators.

LI-7: The platform will not support advanced analytics, such as predictive sales modeling or real-time user behavior tracking, in the initial analytics dashboard.

LI-8: GameHub will not include features for in-game monetization (e.g., in-app purchases or microtransactions) but will focus solely on game and asset sales.

LI-9: The platform will not provide translation services for game descriptions, assets, or interfaces into multiple languages. Developers must provide their own translations.

LI-10: The platform is not designed to provide personal or career development tools such as resume building, mentorship programs, or networking features beyond the basic collaboration tools.

II. Project Management Plan

1. Overview

1.1 Scope & Estimation

No.	WBS Item	Complexity	Est. Effort (man-days)
1.	**Initiation**		**19**
1.1	Define the background and context of the product	Complex	7
1.2	Find problems of users	Medium	3
1.3	Existed system analysis	Simple	2
1.4	Define project scope & limitation	Complex	7
2.	**Planning & Requirement**		**33**
2.1	Initialize projects and setup the development environment	Medium	7
2.2	Setup project management tool	Simple	2
2.3	Develop Project Management Plan Document and Project Tracking Sheet	Complex	10

2.4	Analysis requirements and capture software Software Requirement Specifications (SRS)	Complex	14
3.	**Software Designing**		**30**
3.1	Design High Level Architecture	Complex	10
3.2	Design UI prototypes	Complex	10
3.3	Design conceptual & logical ERD	Complex	10
4.	**Implementation**		**121**
4.1	Implement Authentication Features	Complex	7
4.2	Implement Web application features for Developer role	Complex	30
4.3	Implement Web application features for Designer Role	Complex	21
4.4	Implement Web application for Moderator Role	Complex	14
4.5	Implement Web application for Customer Role	Complex	21
4.6	Implement Web application for Admin Role	Complex	28
5.	**Testing**		**30**
5.1	Uni Test	Complex	10
5.2	Integration Test	Complex	10
5.3	System Test	Complex	10
6.	**Transition**		**28**
6.1	Deploy Final Code	Complex	7
7.	**Closing**		**21**
7.1	Prepare user guides	Complex	12
7.2	Prepare Final Project	Medium	7
7.3	Prepare thesis presentation	Medium	2

Total Estimated Effort (man-days)	282

Table 3 - Scope & Estimation

1.2 Project Objectives

No.	Testing Stage	Test Coverage	No. of Defects	% of Defect	Notes
1	Unit Test	85%	12	30%	
2	Integration Testing	75%	15	37%	
3	System Testing	90%	10	25%	

Table 4 - Project Objectives

Milestone Timeliness (%): 80%

Allocated Effect: 5 (members) * 14 (weeks) * (5 days) = 350 man-days

1.3 Project Risks

#	Risk Description	Impact	Possibility	Response Plans
1	Difficulty in maintaining the codebase	High	Medium	Apply Clean Architecture principles, maintain thorough documentation, and develop necessary system and design diagrams to ensure long-term maintainability.
2	The solution may not align with real-world requirements	High	High	Conduct comprehensive business analysis, gather detailed user feedback, and research similar platforms to ensure the product addresses practical needs.
3	A team member may leave the project due to personal or unforeseen reasons	Critical	Low	Reallocate tasks among remaining team members, establish backup plans, and ensure all members are familiar with shared responsibilities to mitigate impact.

4	Insufficient technical skills for code integration among team members	High	High	Provide focused training sessions, assign mentors within the team, and ensure the use of shared development guidelines and tools to standardize work.

Table 5 - Project Risks

2. Management Approach

2.1 Project Process

- ❖ Scrum Framework

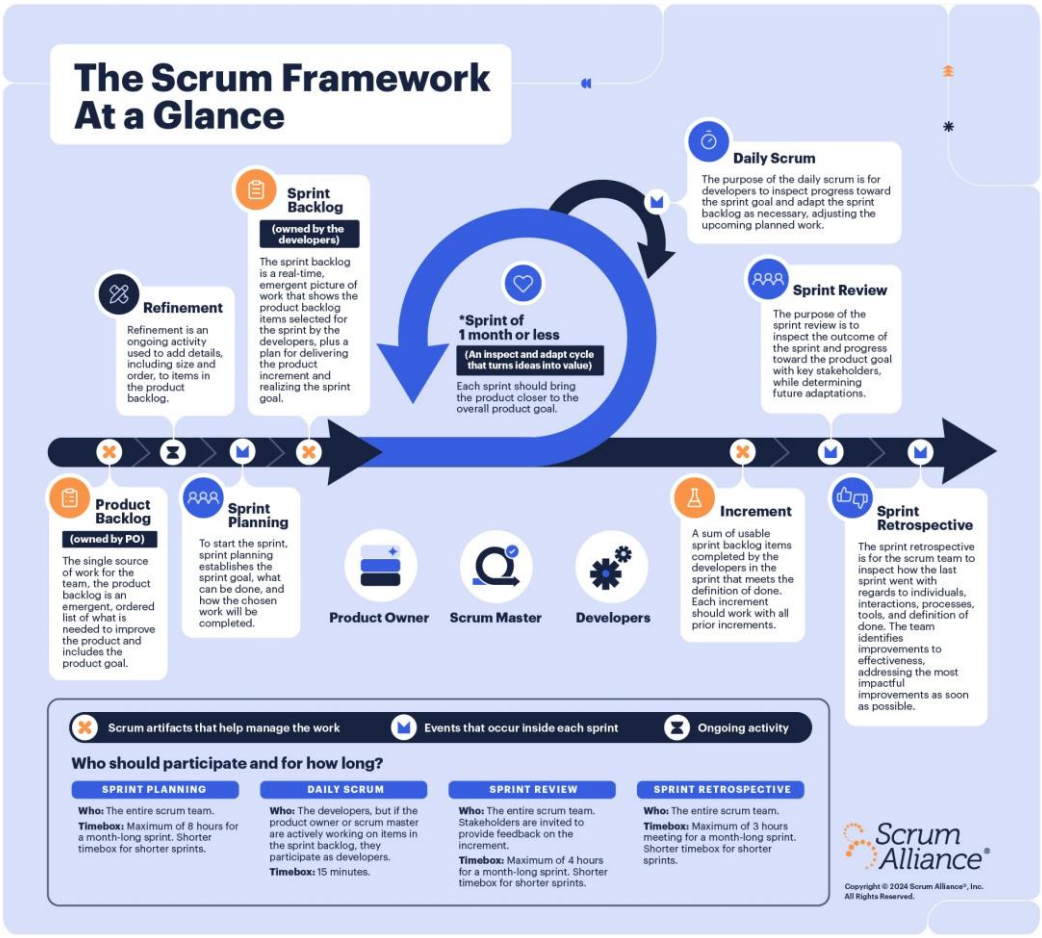

Figure 4 - Scrum Framework

Reference: https://www.scrumalliance.org/about-scrum

GameHub's project management uses the Scrum Framework for software development. Scrum embraces agile principles and relies on cross-functional, self-managing teams to deliver products and services in short cycles, enabling:

- Fast feedback
- Quicker innovation

- Continuous improvement
- Rapid adaptation to change
- Increased customer satisfaction
- Reduced time from idea to delivery

Practicing Scrum offers several advantages, our team can take the following core benefits:

- Minimized risk
- Enhanced ability to manage changing priorities
- Ability to uncover bottlenecks in teams and organizations that get in the way of value delivery
- Increased transparency for stakeholders by delivering working product increments frequently
- Opportunities to inspect and adapt, thereby making course corrections based on what is learned

❖ Trello

Trello is the flexible work management tool where my teams can ideate plans, collaborate on projects, organize workflows, and track progress in a visual, productive, and rewarding way. The key is that we use Trello as our main management tool because it is simple, easy to operate, and especially we can integrate Scrum Framework using the trello management tool.

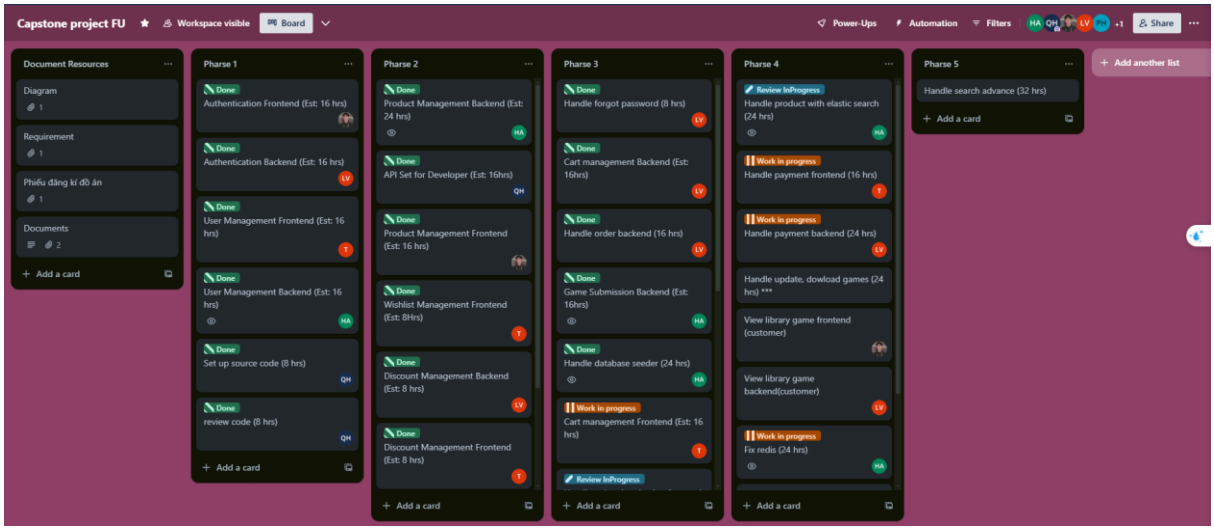

Figure 5 - Trello Management Tool

2.2 Quality Management

The GameHub team is committed to delivering a high-quality software product that meets both user expectations and technical standards. Quality management throughout the project lifecycle is ensured through a combination of process control, code quality assurance, continuous integration, and regular testing activities.

Quality Objectives:

- Ensure that the software functions as expected under defined conditions.

- Maintain high code readability, reusability, and scalability.
- Minimize bugs and performance issues through rigorous testing and review.
- Deliver user-friendly interfaces that enhance user experience (UX).

Key Practices in Quality Management:

Activity	Description
Code Reviews	Team members perform regular peer code reviews to ensure code consistency, adherence to standards, and early detection of bugs.
Automated Testing	Unit tests and integration tests are written for core functionalities to ensure reliability and catch regressions early.
Manual Testing	Functional testing and User Acceptance Testing (UAT) are conducted to verify system behavior against user requirements.
CI/CD Pipeline	Implementing Continuous Integration/Continuous Delivery (CI/CD) ensures code is automatically built, tested, and deployed with minimal human intervention.
Scrum Review & Retrospective	Regular Sprint Reviews and Retrospectives allow the team to inspect the work completed and identify opportunities to improve both product quality and team performance.
Issue Tracking	All issues, bugs, and improvement suggestions are tracked using Trello, allowing for clear task assignment, priority setting, and monitoring progress.

Table 6 - Key Practices in Quality Management

Standards and Tools Used:

- Coding Standards: Follow clean code principles and naming conventions.
- Tools: Trello for project and issue tracking, Git for version control, GitHub Actions (or similar) for CI/CD, VS Code/IDE for development, Postman for API testing.
- Documentation: Maintain clear and concise documentation, including API specs, diagrams, and architecture overviews.

By applying these quality management practices consistently, the GameHub team ensures that the final product is not only functional and robust but also maintainable and aligned with user expectations.

2.3 Training Plan

Training Area	Participants	When, Duration	Waiver Criteria
Nextjs	Trần Thẩm Anh Toàn Bạch Doãn Vương Quách Hoàng Huy	Week 1 - 7 Days	Mandatory
ASP .NET Core, Project Architecture: Clean Architecture	Nguyễn Hoàng Anh Vũ Lê Lâm Hoàng	Week 1, 2 - 12 Days	Mandatory
Project workflow, Git, Github, Sourcetree	Everyone	Week 1 -7 days	Mandatory
Deployment: AWS (Elastic Kubernetes Service, Elastic Container Registry, Elastic Compute Cloud, Simple Storage Bucket, Simple Email Service, Virtual Private Cloud, Elastic Load Balancing, Relational Database Service for MySQL, Route53, Amazon Amplify, Amazon TextTract, API Gateway, AWS Lambda, DynamoDB), Infrastructure as Code (Terraform), CICD Pipeline (Github action)	Bạch Doãn Vương	Week 1, 2 - 14 days	Mandatory
Integration Technologies: Elastic Search, Redis	Nguyễn Hoàng Anh Vũ Lê Lâm Hoàng Quách Hoàng Huy	Week 1, 2 - 14 days	Mandatory

Table 7 - Training Plan

3. Project Deliverables

No.	Deliverable	Start Date	Due Date
1	**Report 1** - Project Introduction	01/01/2025	06/01/2025
2	Project Tracking Sheet	01/01/2025	03/01/2025

3	**Report 2** - Project Management Plan	07/01/2025	12/01/2025
4	**Report 3** - Software Requirement Specification	13/01/2025	22/01/2025
5	**Report 4** - Software Design Document	24/01/2025	07/02/2025
6	**Report 5** - Software Test Document	08/03/2025	17/03/2025
7	**Report 6** - Software User Guides	01/04/2025	13/04/2025
8	**Module Report**	20/01/2025	10/04/2025
9	Web developer modules	20/01/2025	09/02/2025
10	Web customer modules	10/02/2025	24/02/2025
11	Web moderator modules	25/02/2025	11/03/2025
12	Web admin modules	12/03/2025	26/03/2025
13	Web designer modules	27/03/2025	10/04/2025
14	**Final Project Report document**	11/04/2025	17/04/2025
15	Final presentation slide	18/04/2025	20/04/2025

Table 8 - Project Delivery Plan

4. Project Organization

4.1 Team and Structure

Role	Full Name
Product Owner	Nguyễn Thanh Trí
Scrum Master	Quách Hoàng Huy
Developer Team	Vũ Lê Lâm Hoàng
	Nguyễn Hoàng Anh
	Trần Thẩm Anh Toàn
	Bạch Doãn Vương

Table 9 - Team and Structure

4.2 Roles and Responsibilities

Responsibility	Quách Hoàng Huy	Vũ Lê Lâm Hoàng	Nguyễn Hoàng Anh	Trần Thẩm Anh Toàn	Bạch Doãn Vương
Planning & Documents					
Write Project Introduction document	R	I	D	I	I
Write Project Management Plan document	R	D	D	I	I
Write Software Requirement Specification document	R	I	D	D	I
Write Project Tracking document	R	I	D	I	D
Write Software Design Document	D	I	S	D	I
Write Test Document	R	D	D	I	I
Write Software User Guides document	D	I	R	I	D
Write Final Project Report Document	R	I	D	I	I
Database					
Design conceptual & logical ERD	R	D	D	D	D

Prepare data source for the system	R	D	D	I	I
Usecases & Low level designs					
Analyze user requirements into usecases & business rules.	R	D	D	D	D
Draw usecase diagram	R	D	D	D	D
Draw sequence diagrams	R	D	D	D	D
Draw state machine diagrams	R	D	D	D	D
Draw activity diagrams	R	D	D	D	D
Draw package diagram	R	D	D	D	D
Draw class diagram	R	D	D	D	D
Frontend side					
Design UI prototypes	R	I	I	D	D
Develop frontend Web & modules	D	I	I	D	D
Review code frontend	D	I	I	D	D
CI/CD frontend code package	R	I	I	I	D
Backend side					
Develop backend APIs & modules	D	D	D	I	I
Review code backend	D	D	D	I	I
CI/CD backend code package	R	I	I	I	D
Testing					
Write testing plan & test cases	D	D	D	D	D
Do Testing	D	D	D	D	D
Deployment side					
Deploy source code modules on Cloud environment for production release	D	I	I	I	D
Game side					
Create real Game project	I	D	I	I	I

Implement Game Hub APIs on real Game projects	I	D	I	I	I

Table 10 - Roles and Responsibilities

5. Project Communications

Communication Item	Who/Target	Purpose	When, Frequency	Type, Tool, Method(s)
Meeting with Advisor	Supervisor, team members	- Review documentation - Evaluate project progress and result - Technical support	Weekly	Google Meet
Meeting with Team	Team members	- Evaluate task progress and result - Assign tasks - Discuss problems and solutions	Weekly	Google Meet, Zalo
Messages	Team members	- Share daily task progress - Share potential roadblocks	Daily	Zalo

Table 11 - Project Communications

6. Configuration Management

6.1 Document Management

- Google Drive

Our project uses Google Drive - a cloud based storage service that enables users to store and access files online. We take advantage of Google Drive to save meeting reports, records, diagrams, images, designs, and documents and organise each by purpose in designated folders.

6.2 Source Code Management

The GameHub project uses **Git** for version control and **GitHub** for repository hosting and collaboration.

We follow a clear **branching strategy**:

- main: Stable, production-ready code
- develop: Integration of features before release
- feature/*: New features
- bugfix/*, hotfix/*,fix/*: Issue resolution

All changes are made via **pull requests**, reviewed by team members to maintain code quality. **Commit messages** follow a consistent format (e.g., feat(login): add user login API).

Tools Used:

- **GitHub**: Code hosting, pull requests, issue tracking
- **SourceTree**: Visual Git management
- **GitHub Actions** (optional): CI/CD automation

This setup ensures safe, traceable, and collaborative development.

6.3 Tools & Infrastructures

Category	Tools / Infrastructure
Technology	(FrontEnd) TypeScript, NextJS; (Mobile) Flutter; ASP DotNet Core (BackEnd)
Database	MySQL, MongoDB
IDEs/Editors	Visual Studio Code, Visual Studio
Diagramming	Draw.io, Visual-Paradigm
Documentation	Google Docs/Sheets
Version Control	GitHub (Source Codes), Google Drive (Documents)
Deployment server	AWS Web Service
Project management	Trello (Tasks, Defects), Google Sheet (Schedule)

Table 12 - Tools & Infrastructures

III. Software Requirement Specification

1. Product Overview

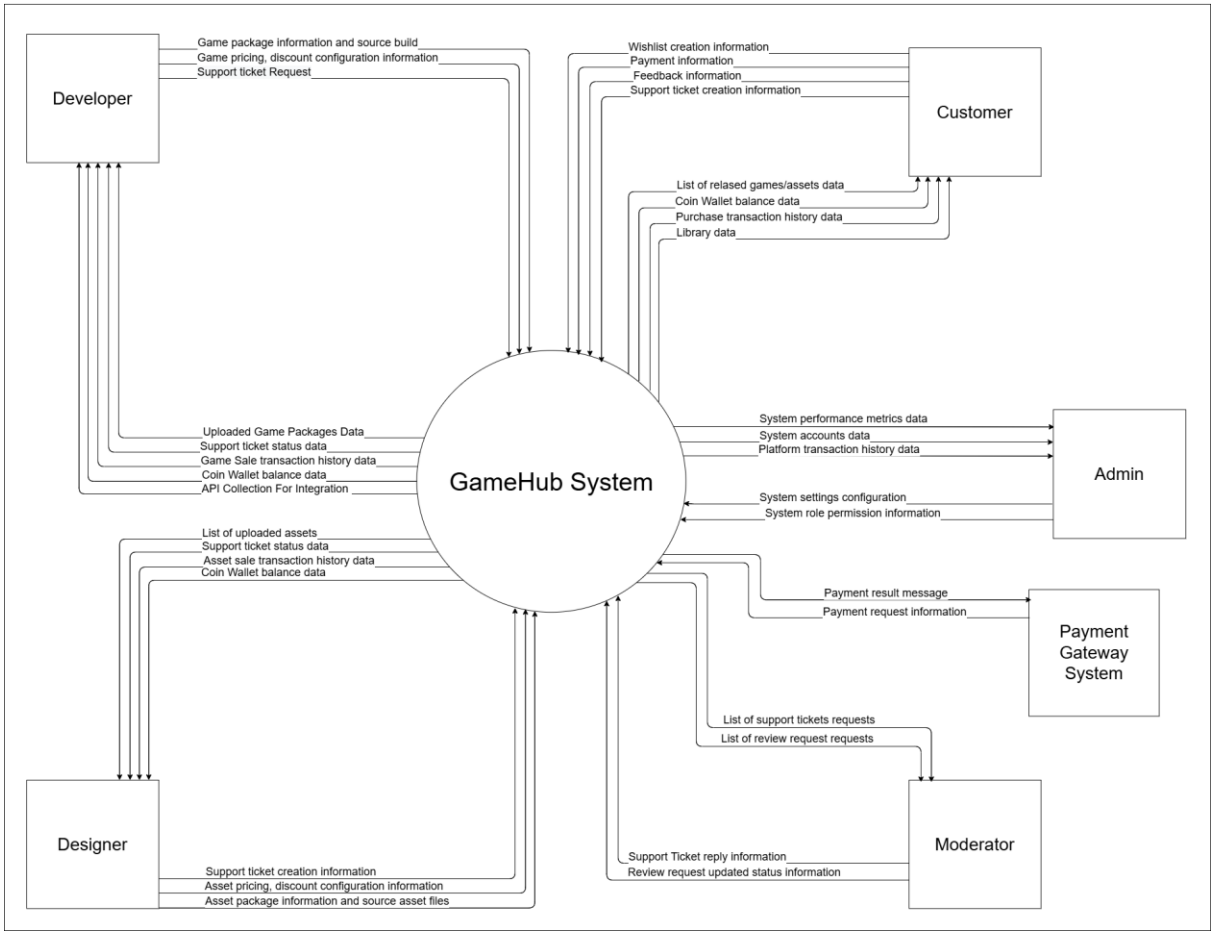

Figure 6 - Product Context Diagram

2. User Requirements

2.1 Actors

No.	Actor	Description

1	**Administrator**	The administrator is responsible for overseeing and maintaining the system. Administrators primarily use the web application to perform the following works: - Access the system with given account and update account profile - Monitoring revenue and financial metrics - Manage revenue sharing and profit allocation - Configure platform settings and preferences - Generate reports and review transaction history - Track system performance and usage metrics - Manage system accounts role and permissions
2	**Moderator**	The moderator is responsible for maintaining the quality, compliance, and integrity of content on the platform. Moderators primarily use the web application to perform following works: - Access the system with given account and update account profile - Review and approve/reject game products and assets - Ban or unban non-compliant content of game products and assets - Handle support tickets and resolve user and developer issues - Track and audit the history tickets and moderation actions - Manage discount promotions

3	**Customer**	The customer is primarily responsible for exploring, purchasing, and engaging with game products and assets on the platform. Customers use the web application to perform the following works: - Create individual account, access the system, and update account profile - Browse, search, and view game products and assets - Manage wishlists and purchased items in the library - Purchase games/assets using coins and run them on devices - Provide feedback and rate games/assets - Buy coins securely and manage wallet balance and transactions - Create and manage support tickets for assistance
4	**Developer**	The developer is responsible for creating, managing, and monetizing game products on the platform. Developers primarily use the web application to perform the following works: - Create individual account, access to the entry system by created account and update account profile - Manage game products profiles and updates - Upload and update game builds - Configure pricing and discounts for game products - Release game products for sale and manage review requests - Track sales performance and revenue - Submit withdrawal requests and manage payments - Submit and manage support tickets' status - Manage projects and GameHub APIs implement in projects

5	**Designer**	The designer is responsible for creating, managing, and monetizing game assets on the platform. Designers primarily use the web application to perform the following works: - Create individual account, access to the entry system by created account and update account profile - Manage game asset profiles and updates - Upload and update game asset files - Configure pricing and discounts for game assets - Release game assets for sale and manage review requests - Track sales performance and revenue - Submit withdrawal requests and manage payments - Submit and manage support tickets' status
6	**Guest**	Guest is the person who has not been identified by the GameHub System. Guest use web application to perform the following works: - Browse, search, and view game products and assets - Register to create account

Table 13 - Actors

2.2 Use Cases

2.2.1 Diagrams(s)

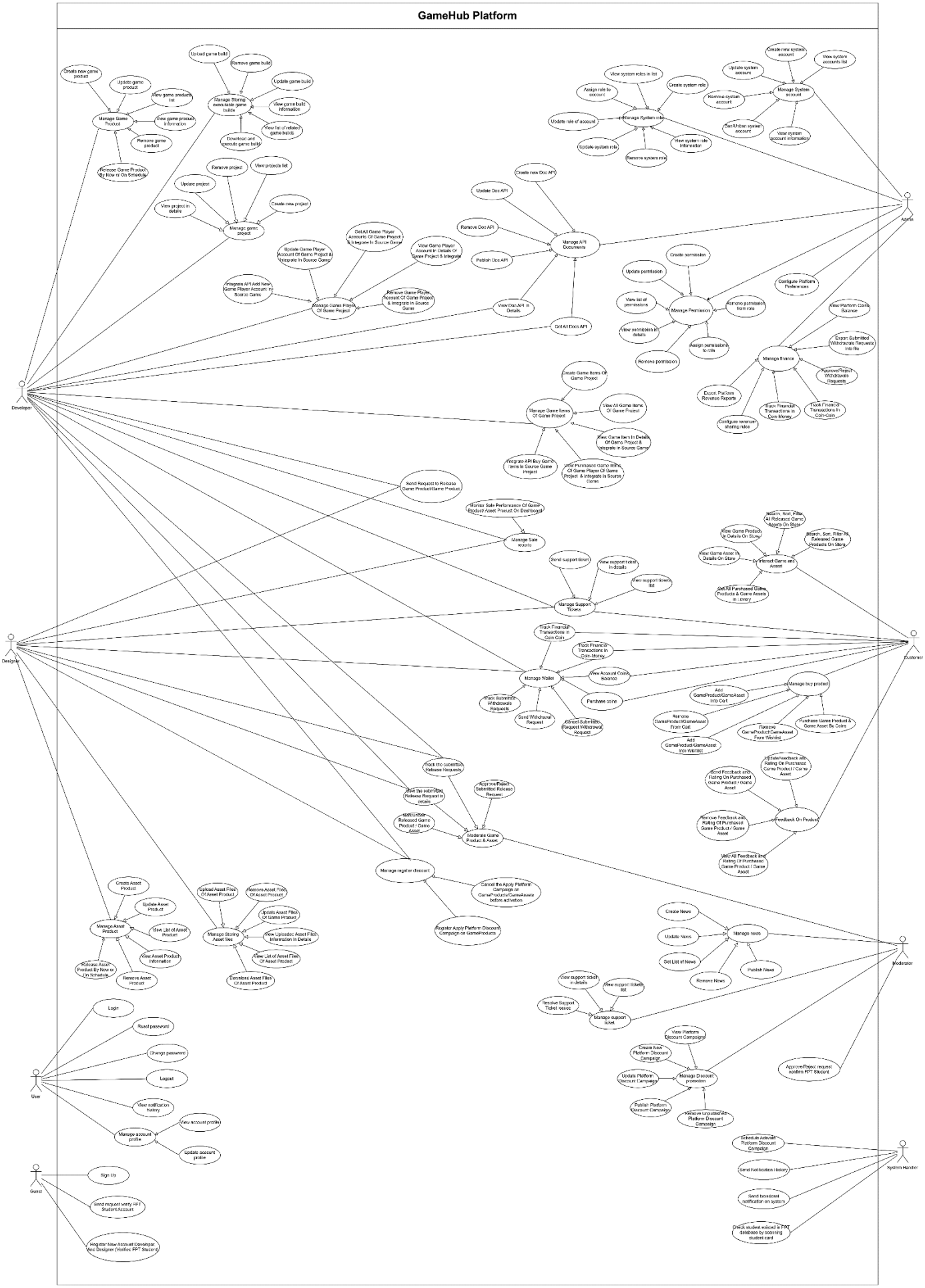
GameHub Platform
Upload game build
Remove game build
Update game build
View game build information
View list of related game builds
Download and execute game build
Manage Storing executable game builds
Create new game product
Update game product
View game products list
View game product information
Remove game product
Manage Game Product
Release Game Product By Now or On Schedule
Create new system account
View system accounts list
Update system account
Remove system account
Manage System account
View system account information
Set Unlock system account
Create system role
View system roles in list
Assign role to account
Update role of account
Update system role
Remove system role
View system role information
Manage System role
Upcase project
Remove project
View projects list
Create new project
View project in details
Manage game project
Create new Doc API
Update Doc API
Remove Doc API
Publish Doc API
Manage API Documents
Get All Game Player Accounts Of Game Project & Integrate In Source Game
Update Game Player Account In Source Game
Integrate API Add New Game Player Account in Source Game
Manage Game Player Of Game Project
View Game Player Account In Details Of Game Project & Integrate
Remove Game Player Account Of Game Project & Integrate In Source Game
View Doc API In Details
Get All Docs API
Create permission
Update permission
View list of permissions
View permission in details
Remove permission
Manage Permission
Remove permission from role
Assign permissions to role
Configure Platform Preferences
View Platform Coins Balance
Export Submitted Withdrawals Requests into file
Approve/Reject Withdrawals Requests
Track Financial Transactions in Coin-Coin
Track Financial Transactions in Coin-Money
Export PlatCoin Revenue Reports
Configure revenue-sharing rules
Manage finance
Create Game Items Of Game Project
View All Game Items Of Game Project
View Game Item In Details Of Game Project & Integrate in Source Game
Integrate API Buy Game Items In Source Game Project
View Purchased Game Items Of Game Player Of Game Project & Integrate In Source Game
Manage Game Items Of Game Project
Send Request to Release Game Product/Game Product
Monitor Sale Performance Of Game Product/ Asset Product On Dashboard
Manage Sale reports
Search, Sort, Filter All Released Game Assets On Store
View Game Product In Details On Store
Search, Sort, Filter All Released Game Products On Store
View Game Asset In Details On Store
Interact Game and Asset
Get All Purchased Game Products & Game Assets in Library
Send support ticket
View support ticket in details
View support tickets list
Manage Support Ticket
Track Financial Transactions in Coin-Coin
Track Financial Transactions in Coin-Money
Manage Wallet
View Account Coins Balance
Purchase coins
Track Submitted Withdrawals Requests
Send Withdrawal Request
Cancel Submitted Request Withdrawal Request
Manage buy product
Add GameProduct/GameAsset into Cart
Remove GameProduct/GameAsset From Cart
Add GameProduct/GameAsset into Wishlist
Remove GameProduct/GameAsset From Wishlist
Purchase Game Product & Game Asset By Coins
Track the submitted Release Requests
View the submitted Release Request in details
Approve/Reject Submitted Release Request
Remove Released Game Product / Game Asset
Moderate Game Product & Asset
Update Feedback and Rating On Purchased Game Product / Game Asset
Send Feedback and Rating On Purchased Game Product / Game Asset
Remove Feedback and Rating Of Purchased Game Product / Game Asset
Feedback On Product
View All Feedback and Rating Of Purchased Game Product / Game Asset
Manage register discount
Cancel the Apply Platform Campaign on GameProducts/GameAssets before activation
Register Apply Platform Discount Campaign on GameProducts
Create Asset Product
Update Asset Product
View List of Asset Product
View Asset Product Information
Remove Asset Product
Release Asset Product By Now or On Schedule
Manage Asset Product
Upload Asset Files Of Asset Product
Remove Asset Files Of Asset Product
Update Asset Files Of Game Product
View Uploaded Asset Files Information In Details
View List of Asset Files Of Asset Product
Download Asset Files Of Asset Product
Manage Storing Asset files
Create News
Update News
Get List of News
Remove News
Publish News
Manage news
View support ticket in details
View support tickets list
Resolve Support Ticket Issues
Manage support ticket
View Platform Discount Campaigns
Create New Platform Discount Campaign
Update Platform Discount Campaigns
Publish Platform Discount Campaign
Manage Discount promotion
Remove Unpublished Platform Discount Campaign
Approve/Reject request confirm FPT Student
Schedule Activate Platform Discount Campaign
Send notification history
Send broadcast notification on system
Check student entered in FPT database by scanning student card
Login
Reset password
Change password
Logout
View notification history
View account profile
Update account profile
Manage account profile
Sign Up
Send request verify FPT Student Account
Register New Account Developer And Designer (Verified FPT Student)
Admin
Developer
Designer
Customer
Moderator
System Handler
User
Guest

Figure 7 - Use Case Diagram

2.2.2 Descriptions

ID	Use Case	Actors	Use Case Description
UC-01	Sign up	Customer	Register a new account on the platform.
UC-02	Login	Customer, Developer, Designer, Admin, Moderator	Authenticate to access personal or platform-specific functionalities.
UC-03	Reset Password	Customer, Developer, Designer	Reset the account password when forgotten.
UC-04	Change Password	Customer, Developer, Designer	Update the account password.
UC-05	Log out	All Roles	Sign out from the current session.
UC-06	View Account Profile	Customer, Developer, Designer, Admin	View profile and account details.
UC-07	Update Account Profile	Customer, Developer, Designer, Admin	Edit personal profile and account settings.
UC-08	Approve/Reject Request Confirm FPT Student	Moderator	Approve or reject requests for FPT student verification.
UC-09	Check Student Existed In FPT Database By Scanning Student Card	System Handler	Automatically check FPT database for student existence using student card.
UC-10	Send request verify FPT Student Account	Guest	Request verification as an FPT student for platform privileges.
UC-11	Register New Account Developer, And Designer (Verified FPT Student)	Guest	Register as Developer or Designer after verifying FPT student status.
UC-12	Send Broadcast Notification On System (Broadcast)	System, Admin	Broadcast a notification to all or many users.
UC-13	View Notification History	All Roles	Check history of received notifications.

UC-14	Send Notification through Email	System	Send email notifications to users.
UC-15	View System Accounts List	Admin	List all system user accounts.
UC-16	Create New System Account	Admin	Create a new system-level user account.
UC-17	Update System Account	Admin	Modify an existing system account.
UC-18	Remove System Account	Admin	Delete a system account.
UC-19	Ban/Unban System Account	Admin	Restrict or restore access to a system account.
UC-20	View System Account Information	Admin	View details of a specific system account.
UC-21	View System Roles In Lists	Admin	Display all defined roles in the system.
UC-22	Create System Role	Admin	Add a new role with specific permissions.
UC-23	Update System Role	Admin	Modify details or permissions of a role.
UC-24	Remove System Role	Admin	Delete a defined system role.
UC-25	View System Role Information	Admin	See detailed information of a role.
UC-26	Assign role to account	Admin	Assign a specific role to a user account.
UC-27	Update role of account	Admin	Change the role assigned to an account.
UC-28	Create Permission	Admin	Define a new permission in the system.
UC-29	Update Permission	Admin	Modify an existing permission.
UC-30	Remove Permission	Admin	Delete a permission from the system.
UC-31	Assign Permissions to Role	Admin	Add permissions to a role.
UC-32	Remove Permission from Role	Admin	Revoke permissions from a role.

UC-33	View List of Permissions	Admin	List all permissions available in the system.
UC-34	View Permission in Details	Admin	Display details of a specific permission.
UC-35	Configure Platform Preferences	Admin	Set global configurations and settings for the platform.
UC-36	Create new Doc API	Moderator	Add new API documentation for developers.
UC-37	Update Doc API	Moderator	Edit existing API documentation.
UC-38	Remove Doc API	Moderator	Delete API documentation entries.
UC-39	Publish Doc API	Moderator	Make API documentation publicly visible.
UC-40	View Doc API In Details	Moderator, Developer	See full content of a specific API documentation.
UC-41	Get All Docs API	Moderator, Developer	Retrieve a list of all documented APIs.
UC-42	Create New Project	Developer	Start a new game development project.
UC-43	Update Project	Developer	Edit project information and settings.
UC-44	Remove Project	Developer	Delete an existing project.
UC-45	View List of Projects	Developer	See all projects created by the developer.
UC-46	View Project in Details	Developer	Open a project and view all its detailed info.
UC-47	Integrate API Add New Game Player Account In Source Game	Developer	Use API to create a new game player account in an external game system.
UC-48	Integrate API Add New Game Player By Using GameHub Account	Developer	Use GameHub account to create a game player in an external game system.
UC-49	Get All Game Player Accounts Of Game Project & Integrate In Source Game	Developer	Retrieve and integrate all player accounts for a game project.

UC-50	View Game Player Account In Details Of Game Project & Integrate	Developer	View detailed info of a player account and integrate it in the source game.
UC-51	Update Game Player Account Of Game Project & Integrate In Source Game	Developer	Modify a player account and sync changes to the game system.
UC-52	Remove Game Player Account Of Game Project & Integrate In Source Game	Developer	Delete a game player account and reflect change in source game.
UC-53	Create Game Items Of Game Project	Developer	Create new in-game items within a game project.
UC-54	View All Game Items Of Game Project	Developer	View the list of all created game items for a project.
UC-55	View Game Item In Details Of Game Project & Integrate in Source Game	Developer	View and integrate item info in source game logic.
UC-56	View Purchased Game Items Of Game Player Of Game Project & Integrate in Source Game	Developer	See items purchased by a game player.
UC-57	Integrate API Buy Game Items In Source Game Project	Developer	Enable players to purchase items through API integration.
UC-58	Create New Game Product	Developer	Publish a new game product for sale.
UC-59	Update Game Product	Developer	Edit information of an existing game product.
UC-60	View Game Products List	Developer	Browse the developer’s list of created game products.
UC-61	View Game Product Information	Developer	View details of a selected game product.
UC-62	Remove Game Product	Developer	Delete a published or draft game product.
UC-63	Upload Game Builds Of Game Product	Developer	Upload game builds (e.g., executables, packages) for a product.
UC-64	Remove Game Builds Of Game Product	Developer	Delete uploaded builds for a game product.
UC-65	Update Game Builds Of Game Product	Developer	Replace or edit information of a game build.

UC-66	View Game Build Information In Details	Developer	Inspect detailed data of a game build.
UC-67	View List of Related Game Builds Of Game Product	Developer	View all builds associated with a specific game product.
UC-68	Download Game Build Of Game Product	Developer, Moderator, Customer	Download the game files of a released game build.
UC-69	Release GameProduct By Now or On Schedule	Developer	Release a game product immediately or set a future release date.
UC-70	Create Asset Product	Designer	Add a new asset (e.g., models, textures) to the asset library.
UC-71	Update Asset Product	Designer	Modify the details of an existing asset.
UC-72	View List of Asset Product	Designer	View all game assets uploaded by the designer.
UC-73	View Asset Product Information	Designer	View detailed information about a specific asset.
UC-74	Remove Asset Product	Designer	Delete a game asset from the library.
UC-75	Upload Asset Files Of Asset Product	Designer	Upload files (textures, models, etc.) to an asset entry.
UC-76	Remove Asset Files Of Asset Product	Designer	Delete one or more files associated with an asset.
UC-77	Update Asset Files Of Game Product	Designer	Modify asset file metadata or replace files.
UC-78	View Uploaded Asset Files Information In Details	Designer	View the properties of an uploaded asset file.
UC-79	View List of Asset Files Of Asset Product	Designer	Browse all files attached to a specific asset.
UC-80	Download Asset Files Of Asset Product	Designer, Moderator, Customer	Download individual asset files.
UC-81	Release Asset Product By Now or On Schedule	Designer	Release a complete game package now or at a scheduled time.
UC-82	Send Request to Release Game Product/Game Product	Developer, Designer	Submit a request to release a game product or asset.

UC-83	Approve/Reject Submitted Release Request	Moderator	Approve or reject submitted release requests.
UC-84	Track the submitted Release Requests	Moderator, Developer, Designer	View the status of submitted release requests.
UC-85	View the submitted Release Request in details	Moderator, Developer, Designer	View detailed info about a release request.
UC-86	Ban/Unban Released Game Product / Game Asset	Moderator	Block or unblock a released game product or asset.
UC-87	View Platform Discount Campaigns	Moderator	See all discount campaigns running on the platform.
UC-88	Create New Platform Discount Campaign	Moderator	Create a new platform-level discount campaign.
UC-89	Update Platform Discount Campaign	Moderator	Edit details or rules of a discount campaign.
UC-90	Publish Platform Discount Campaign	Moderator	Make the discount campaign active and visible to users.
UC-91	Publish Platform Discount Campaign	Moderator	View in-depth information about a specific campaign.
UC-92	Remove Unpublished Platform Discount Campaign	Moderator	Delete a draft campaign that has not been published.
UC-93	Schedule Activate Platform Discount Campaign	System Handler	Set a start time for campaign activation.
UC-94	Register Apply Platform Discount Campaign on GameProducts	Developer, Designer	Register a product/asset to use a discount campaign.
UC-95	Cancel the Apply Platform Campaign on GameProducts/GameAssets before activation	Developer, Designer	Cancel campaign application before it goes live.
UC-96	Search, Sort, Filter All Released Game Products On Store	Customer	Browse released game products using search, sort, and filter options.
UC-97	Search, Sort, Filter All Released Game Assets On Store	Customer	Browse released game assets using search, sort, and filter options.
UC-98	View Game Product In Details On Store	Customer	View complete details of a game product listed on the store.

UC-99	View Game Asset In Details On Store	Customer	View complete details of a game asset listed on the store.
UC-100	Add GameProduct/GameAsset Into Cart	Customer	Add selected product or asset to shopping cart.
UC-101	Remove GameProduct/GameAsset From Cart	Customer	Remove product or asset from shopping cart.
UC-102	Add GameProduct/GameAsset Into Wishlist	Customer	Add product or asset to personal wishlist.
UC-103	Remove GameProduct/GameAsset From Wishlist	Customer	Remove product or asset from wishlist.
UC-104	Purchase Coins	Customer	Buy coins to use for purchases on the platform.
UC-105	View Account Coins Balance	Developer, Designer, Customer	View remaining coin balance in personal account.
UC-106	View Platform Coins Balance	Admin	View the total coins in the platform across all users.
UC-107	Purchase Game Product & Game Asset By Coins	Customer	Use coins to purchase a game product or asset.
UC-108	Get All Purchased Game Products & Game Assets In Library	Customer	View and manage purchased items in personal library.
UC-109	Send Feedback and Rating On Purchased Game Product / Game Asset	Customer	Submit a review and rating for a purchased product or asset.
UC-110	Remove Feedback and Rating Of Purchased Game Product / Game Asset	Customer	Delete a submitted feedback or rating.
UC-111	View All Feedback and Rating Of Purchased Game Product / Game Asset	Customer	Browse all reviews and ratings for a product or asset.
UC-112	Update Feedback and Rating Of Purchased Game Product/ Game Asset	Customer	Update a review and rating for a purchased product or asset of the owner.
UC-113	Configure Revenue-Sharing Rules	Admin	Set rules for revenue distribution between platform and creators.

UC-114	Export Platform Revenue Reports	Admin	Export revenue data for reporting or analysis.
UC-115	Monitor Sale Performance Of Game Product/ Asset Product On Dashboard	Developer, Designer	Track how well products/assets are selling.
UC-116	Track Financial Transactions In Coin-Coin	Developer, Designer, Admin, Customer	Monitor transactions between users using coins.
UC-117	Track Financial Transactions In Coin-Money	Developer, Designer, Admin, Customer	Monitor money transactions involving coin conversion.
UC-118	Send Withdrawal Request	Developer, Designer	Submit a request to withdraw earned coins into money.
UC-119	Cancel Submitted Request Withdrawal Request	Developer, Designer	Cancel a previously submitted withdrawal request.
UC-120	Track Submitted Withdrawals Requests	Admin, Developer, Designer	View the status of all submitted withdrawal requests.
UC-121	Export Submitted Withdrawals Requests into file	Admin	Export withdrawal data to external files.
UC-122	Approve/Reject Withdrawals Requests	Admin	Review and take action on withdrawal requests.
UC-123	Create News	Moderator	Create a new news post or announcement.
UC-124	Update News	Moderator	Edit an existing news post.
UC-125	Get List of News	Moderator	View the list of all platform news posts.
UC-126	Remove News	Moderator	Delete an old or irrelevant news post.
UC-127	Publish News	Moderator	Make a news post publicly visible.
UC-128	View support tickets list	Moderator, Customer, Developer, Designer	Browse all support tickets submitted to the platform.
UC-129	View support ticket in details	Moderator, Customer,	View full information of a support ticket.

		Developer, Designer	
UC-130	Resolve Support Ticket Issues	Moderator	Provide resolutions for reported support issues.
UC-131	Send support ticket	Developer, Customer, Designer	Submit a support request for assistance.

Table 14 - Use Cases

3. Functional Requirements

3.1 System Functional Overview

3.1.1 Screen Flow

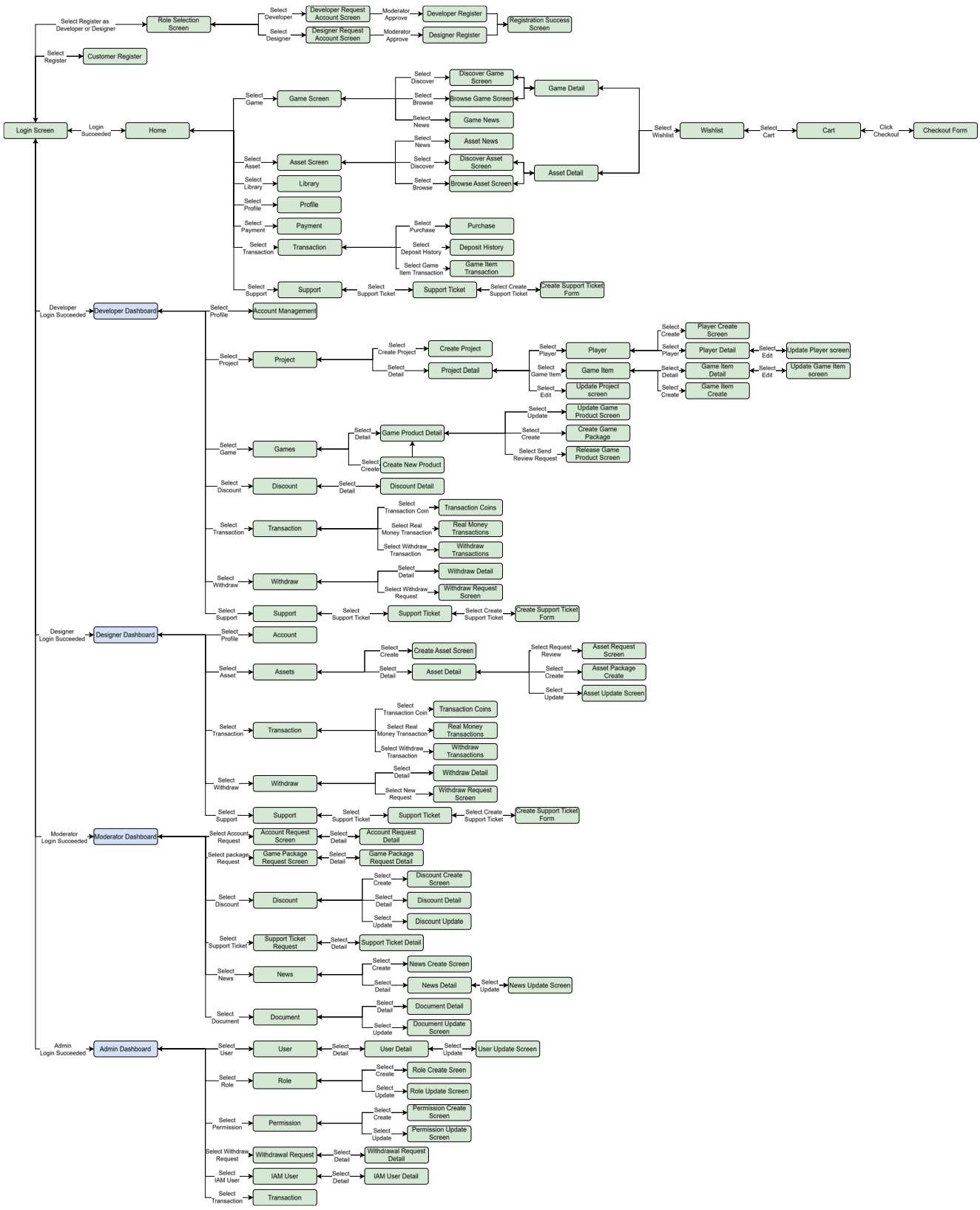

Figure 8 - Screen Flow Diagram

3.1.2 Screen Description

Example:

#	Feature	Screen	Description

1	Authentication	Login screen	This screen allows users to log in to their account, or sign up if they don't have one.
2	Register	Customer register screen	This screen allows users to register an account with the customer role.
3	Register	Developer register screen	This screen is for users who want to become a developer. They need to provide a student ID card, then wait for a moderator to verify it before proceeding to step 2, where they will enter basic fields such as username, password, and confirm password.
4	Register	Designer register screen	This screen is for users who want to become a designer. They need to provide a student ID card, then wait for a moderator to verify it before proceeding to step 2, where they will enter basic fields such as username, password, and confirm password.
5	Home	Home screen	On the home page, customers can view an introduction to the website as well as the About Us section. There are buttons that navigate users to pages such as Game and Asset. Users are not required to log in on this page.
6	Game discover	Game discover screen	On this page, users can view newly released games, trending games, and highly rated games. Users are not required to log in on this page.
7	Game browse	Game browse screen	On the game browse page, users can explore all the games available on the system and use filters to find the games they want. Users are not required to log in on this page.
8	Game news	Game news screen	On the game news page, users can view the latest articles, game reviews, and information about upcoming game releases. Users are not required to log in on this page.
9	Asset discover	Asset discover screen	On this page, users can view newly released assets, trending assets, and highly rated assets. Users are not required to log in on this page.
10	Asset browse	Asset browse screen	On the assets browse page, users can explore all the assets available on the system and use filters to find the assets they want. Users are not required to log in on this page.
11	Asset news	Asset news screen	On the assets news page, users can view the latest articles, assets reviews, and information about upcoming assets releases. Users are not required to log in on this page.
12	Game detail	Game detail	On the game detail screen, users can view

		screen	detailed information about the game, including basic info, and they can add the game to their cart or wishlist from here. Users are not required to log in on this page.
13	Asset detail	Asset detail screen	On the asset detail screen, users can view detailed information about the asset , including basic info, and they can add the asset to their cart or wishlist from here. Users are not required to log in on this page.
14	Wishlist	Wishlist screen	On the wishlist page, customers can create folders to organize games or assets. Basic features include adding, deleting, and editing wishlist folders, as well as adding or removing items from the wishlist.
15	Cart	Cart screen	On the cart page, customers can remove items from the cart or proceed to checkout.
16	Checkout	Checkout form	In this form, customers will review the items in the cart and then proceed to payment, purchasing all the items in the cart.
17	Profile	Profile screen	On the profile page, users can view their current account information and make edits if needed.
18	Payment	Payment screen	On the payment page, users can view available top-up amounts, select a suitable amount, and proceed with the payment. The money will be transferred to their account immediately after completion.
19	Customer transaction	Customer transaction screen	On the Customer Transaction page, customers can view Purchase, which contains payment invoices for games or assets they have bought; Deposit History, which shows transactions made when topping up money into the system; and Game Item Transaction, where they can see the game items they have purchased in games.
20	Support	Support screen	On the support page, users can view available FAQ questions and answers. If they still have inquiries, they will be redirected to the support ticket page.
21	Support ticket	Support ticket screen	On the support ticket page, users can view all the tickets they have created, track the status of each ticket, and create new tickets.
22	Create ticket	Ticket form	In this form, users will enter basic fields such as subject, category, description, and attachment file, then submit the ticket and wait for a response.

23	Project management	Project screen	On the project page, developers can view a list of projects and have the ability to add, delete, edit, and update them.
24	Project detail	Project detail	In the project detail page, developers can view the key provided by the system to access the APIs, as well as manage game items and players.
25	Player	Player screen	This page allows basic management of players, including adding, deleting, and editing players.
26	Game Item	Game Item screen	This page allows management of game items, including adding, deleting, editing game items, and viewing details of each item.
27	Update Project	Update Project screen	This page allows updating project information, including editing the name, description, and other related project details.
28	Player Create	Player Create Screen	This page allows creating a new player, including entering basic information such as name, and other details.
29	Player Detail	Player Detail screen	This page displays detailed information of a player, including name, status, and other related details of the player.
30	Game Item Detail	Game Item Detail screen	This page displays detailed information about a game item sold on the website, including the name, description, price, and other related details of the item.
31	Game Item Create	Game Item Create screen	This page allows creating a new game item, including entering the name, description, price, and other necessary details to add the item to the system.
32	Update Player	Update Player screen	This page allows updating player information, including editing the name, age, status, and other related details.
33	Update Game Item	Update Game Item screen	This page allows updating game item information, including editing the name, description, price, and other related details of the item.
34	Game management	Game management screen	In the game management screen, developers can add, delete, and edit games.
35	Game detail	Game management detail screen	In the game detail page under game management, developers can update the game product, add, delete, or edit game packages, and create requests to release the game.
36	Update Game Product	Update Game Product Screen	This page allows updating game product information, including editing the name,

			description, and other related details of the game product.
37	Create Game Package	Create Game Package screen	This page allows creating game packages within a game product, including uploading game builds and entering related details such as name, description, price, and other information for each package.
38	Release Game Product	Release Game Product Screen	This page allows releasing a game product, including confirming and finalizing the release process of the game product after the information and packages have been updated.
39	Create New Product	Create New Product screen	This page allows creating a new product, including entering the name, description, related information, and other necessary details to add the product to the system.
40	Discount	Discount screen	This page allows developers to view the list of game packages that can have discounts applied
41	Discount Detail	Discount Detail screen	This page allows developers to view the discounts that can be applied to the selected game package, and also displays the history of discounts that have been applied to this package.
42	Transaction Coins	Transaction Coins	This page displays coin-related transactions, including game sales and game item sales transactions.
43	Real Money Transactions	Real Money Transactions	This page displays real money transactions, including withdrawal transactions and subscription purchases.
44	Withdraw Transactions	Withdraw Transactions	This page displays withdrawal transactions, including information about the withdrawn amount, recipient, and transaction status.
45	Withdraw	Withdraw screen	This page allows tracking list withdrawal transactions, including the status and details of each withdrawal transaction.
46	Withdraw detail	Withdraw detail screen	This page displays the details of a withdrawal transaction, including information about the withdrawn amount, recipient, status, and other related details of the transaction.
47	Withdraw Request	Withdraw Request Screen	This page allows users to submit withdrawal requests, including entering bank account number, recipient name, amount (minimum 100 coins), bank name, and a note.
48	Assets management	Assets management	This page allows designers to manage their assets, including tracking, adding, deleting, and updating

			asset information in the system.
49	Create Asset Screen	Create Asset Screen	This page allows creating a new asset, including entering the name, description, and other related details of the asset.
50	Asset Detail	Asset Detail	This page displays the details of an asset, including information such as name, description, status, and other related details of the asset.
51	Asset Request	Asset Request Screen	This page allows designers to submit requests for asset package release
52	Asset Package Create	Asset Package Create screen	This page allows creating a new asset package, including entering details such as name, description, and uploading assets into the package.
53	Asset Update Screen	Asset Update Screen	This page allows updating asset information, including editing the name, description, and other related details of the asset.
54	FPT Requests	List of FPT Student Requests	This screen displays all student requests submitted via the FPT system. From here, Moderators can take action on specific requests. Includes the Approve/Reject Request Modal, which allows quick decision-making through a confirmation popup with necessary fields and validations.
55	Doc API Management	List of Doc APIs	This screen shows all existing API documentation created by users or the system. It supports full lifecycle management via: – Create Doc API Screen (for adding new entries), – Update Doc API Screen (for editing), – Doc API Details Screen (for viewing), – Publish Doc API Modal, and – Remove Doc API Modal (for unlisting or deletion).
56	Game Builds	Download Game Build Screen	Moderators and authorized users can download finalized builds of games for testing, verification, or manual deployment. The screen may include version selection and build notes.
57	Game Assets	Download Game Asset Files Screen	This screen allows downloading of raw or processed asset files such as images, audio, animations, etc. It supports filters like asset type and upload date.
58	Release Requests	List of Release Requests	This interface provides an overview of all submitted game release requests. The View Request Details screen offers in-depth information about each submission, and the Approve/Reject Release Modal lets Moderators

			finalize decisions.
59	Released Items	List of Released Items	Lists all publicly available games and assets. Moderators can access the Ban/Unban Released Item Modal to manage visibility or compliance status of listed items.
60	Discount Campaigns	Platform Discount Campaign List	Shows a list of discount campaigns currently active or scheduled. Includes: – Campaign Details Screen, – Create/Update Campaign Screens, – Publish/Remove/Schedule Modals for managing campaign lifecycle.
61	Reports	List of Reports	This screen displays submitted user reports related to content or behavior. The Report Details screen reveals full information, including attached media and timestamps.
62	News management	News management	This page allows managing news, including adding, deleting, editing, and tracking news articles in the system.
63	News Detail	News Detail screen	This page displays the details of a news article, including the title, description, content, and other related details of the article.
64	News Create	News Create Screen	This page allows creating a new news article, including entering the title, description, content, and other related details of the article.
65	Support Tickets management	Support Ticket List	Displays all support requests from users.
66	Support Ticket Details	Support Ticket Details	Displays detailed information about each support ticket, including sender details, issue description, and current status.
67	Resolve Ticket	Resolve Ticket Modal	Allows moderators to take actions such as closing the ticket or following up on the ticket resolution.
68	User management	User management	This page allows the admin to manage users, including viewing account information, editing user details, and toggling the account status (activating or deactivating the account).
69	User Detail	User Detail screen	This page displays detailed information of a user, including name, email, role, account status, and other related details.
70	User Update	User Update Screen	This page allows updating user information, including editing the name, email, role, and account status.
71	Role management	Role management	This page allows managing roles within the system, including creating, editing, and deleting

			roles.
72	Role Create	Role Create Screen	This page allows creating a new role, including entering the role name.
73	Role Update	Role Update Screen	This page allows updating role information, including editing the role name.
74	Permission management	Permission management	This page allows managing permissions within the system, including creating, editing, and deleting permissions.
75	Permission Create	Permission Create Screen	This page allows creating a new permission, including entering the permission name and description.
76	Permission Update	Permission Update Screen	This page allows updating permission information, including editing the permission name and other related details.
77	Role management	Role management	This page displays the basic roles of the website, containing only the role names, without associated permissions.
78	Role Create	Role Create Screen	This page allows creating a new role, including entering the role name.
79	Role Update	Role Update Screen	This page allows updating role information, including editing the role name.
80	IAM User	IAM User	This page allows assigning roles to accounts in the system, including assigning a role to a user.
81	IAM User Detail	IAM User Detail	This page displays detailed information about an IAM user, including name, email, assigned role, and other related details.
82	Withdrawal Request	Withdrawal Request	This page allows the admin to view withdrawal requests and then export the data to transfer the money to the developer or designer.
83	Withdrawal Request Detail	Withdrawal Request Detail	This page displays detailed information about a withdrawal request, including the amount, requester, status, and other related details.
84	Admin transaction	Admin transaction	This page allows the admin to view the revenue-sharing transactions between the platform and developers or designers, including tracking the split amount and related details.

Table 15 - ScreenFlow Description

3.1.3 Non-screen Functions Description

#	Feature	System Function	Description
1	Authentication and Authorization	JWT Token Generation	Generates JSON Web Tokens (JWT) for secure authentication and authorization, allowing users to access protected resources.
2	Authentication and Authorization	Token Validation	Validates JWT tokens on protected routes to ensure users have proper permissions to access resources.
3	Authentication and Authorization	Role-Based Access Control	Implements role-based access control to restrict access to specific routes based on user roles (Customer, Developer, Designer, Moderator, Admin).
4	Security	Cookie Management	Manages secure HTTP-only cookies for storing authentication tokens and preventing XSS attacks.
5	Security	CORS Configuration	Configures Cross-Origin Resource Sharing to control which domains can access the API.
6	Security	Data Encryption	Encrypts sensitive payment history and account money-related data
7	API Integration	Payment Gateway Integration	Processes payment transactions through third-party payment gateways and handles callbacks.
8	API Integration	File Storage Service	Manages file uploads, storage, and retrieval for game packages, assets, and user content.
9	API Integration	Video Thumbnail Generation	Automatically generates thumbnails for video content uploaded to the platform.
10	Notification Service	Email Notifications	Sends email notifications for important events or updates like account creation, purchase confirmation, and review status changes.
11	Notification Service	In-App Notifications	Delivers real-time notifications to users about system events, updates, and interactions.
12	Background Job Processing	Job Scheduling	Schedules background jobs using Hangfire for tasks like sending notifications, deactivating contracts, or other asynchronous processes.
13	Background Job Processing	Queue Management	Manages job queues to ensure proper execution order and prevent system overload.
14	Background Job Processing	Logging	Logs job execution details, errors, and relevant information for monitoring and troubleshooting.
15	Data Management	Database Transactions	Ensures data integrity through atomic transactions when performing complex

			operations across multiple database tables.
16	Data Management	Data Validation	Validates incoming data against predefined schemas before processing or storing.
17	Data Management	Cache Management	Implements caching strategies to improve performance and reduce database load.
18	Workflow Management	State Tracking	Tracks and manages the state of review workflows for game packages and assets.
19	Workflow Management	Transition Management	Handles transitions between different states in approval workflows.
20	Workflow Management	History Recording	Records history of workflow state changes for auditing and tracking purposes.
21	Error Handling	Global Error Handling	Captures and processes unhandled exceptions throughout the application.
22	Error Handling	Sentry Integration	Sends error reports to Sentry for monitoring and debugging.
23	Error Handling	Error Logging	Logs detailed error information for troubleshooting and system improvement.
24	Analytics	User Activity Tracking	Tracks user interactions and activities for analytics and personalization.
25	Analytics	Performance Monitoring	Monitors system performance metrics to identify bottlenecks and optimization opportunities.
26	Analytics	Usage Statistics	Collects and processes usage statistics for business intelligence and reporting.
27	Content Management	Content Validation	Validates user-generated content against platform policies before publishing.
28	Content Management	Media Processing	Processes uploaded media files (images, videos) for optimization and compatibility.
29	Content Management	Version Control	Manages different versions of game packages and assets.
30	System Integration	Webhook Processing	Processes incoming webhooks from third-party services and triggers appropriate actions.

Table 16 - Non-screen Functions Description

3.1.4 Entity Relationship Diagram

3.1.4.1 Diagram

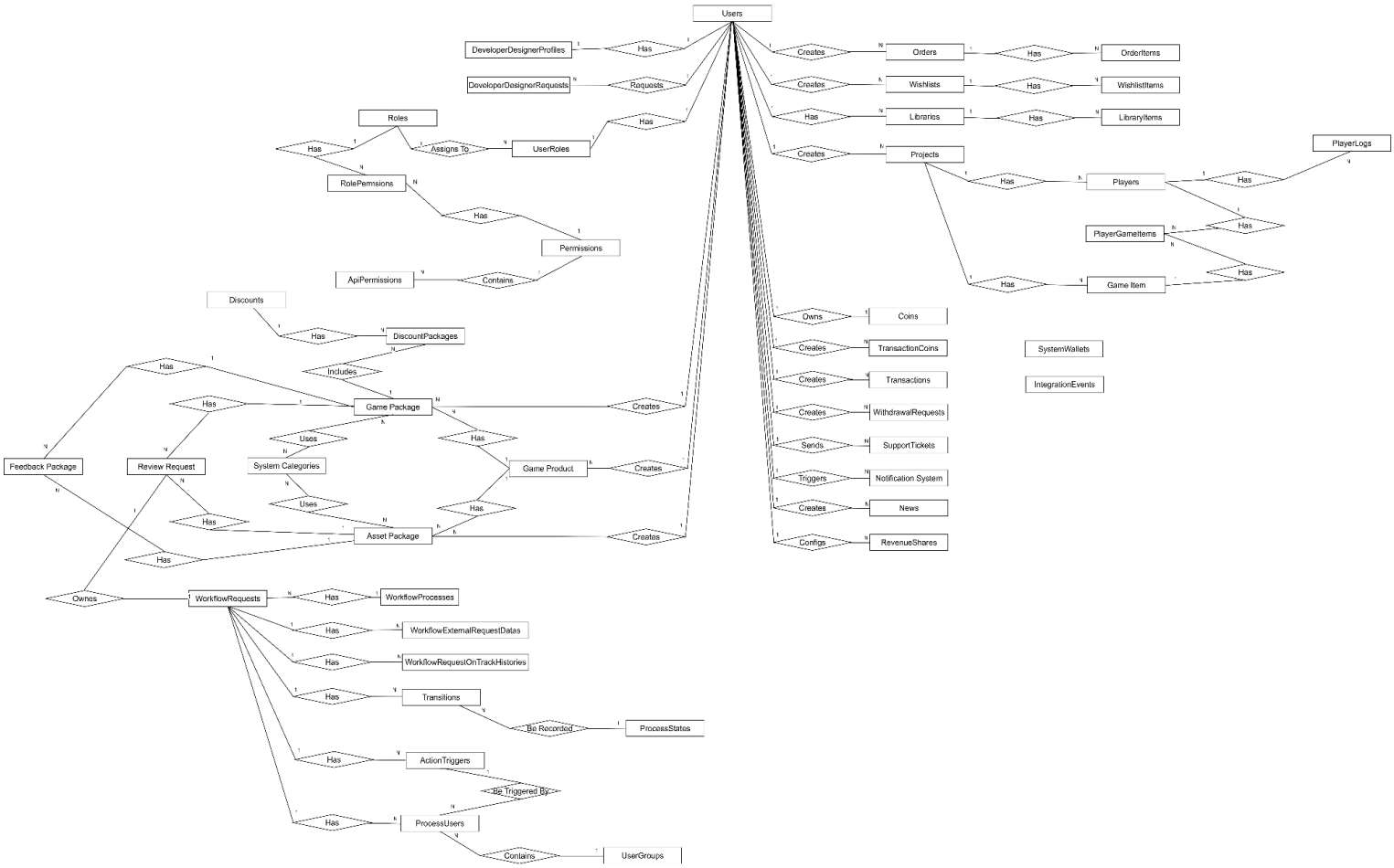

Figure 9 - Entity Relationship Diagram

3.1.4.2 Entities Description

No.	Entity	Description
1	Users	An entity represents a system user, storing authentication and profile information
2	DeveloperDesignerProfiles	An entity represents a user registered to become the system developer, designer, storing profile information
3	DeveloperDesignerRequests	An entity represents an application submitted by a user to register as a system Developer or Designer, storing the approval status and processing details
4	Discounts	An entity represents a price reduction applied to packages or services, storing the discount value, activation time, and related details

5	DiscountPackages	An entity represents a discount applied to a package, storing the discount value and application status information
6	IntegrationEvents	An entity represents an event stored to track its publishing status and related delivery information.
7	News	An entity represents a news item, storing its content, title, and publication information
8	Orders	An entity represents a customer order, storing the buyer information, total payment, and payment status
9	OrderItems	An entity represents an item in an order, storing the item price and related information
10	Wishlists	An entity represents a customer wishlist, storing its details and associated information
11	WishlistItems	An entity represents an item saved in a wishlist, storing its details and related information
12	Libraries	An entity represents a customer's library, storing its details and associated information
13	LibraryItems	An entity represents an item in a customer's library, such as a purchased game package or asset package, storing its associated information
14	Projects	An entity represents a project that provides developers with an unique API connection key, storing the key and related metadata. It is used to enable developers to integrate and call external game APIs within their game source code.
15	Players	An entity represents an individual interacting with the developer's game, linked to the project entity that provides the API key, storing the player's profile information
16	PlayerLogs	An entity represents a log of the time period during which a player engages with the developer's game, storing the start time, end time, and duration of each play session
17	PlayerGameItems	An entity represents a game item within a developer's game project, linked to the Project entity that provides the API key, storing the item's price and related information.
18	WorkflowProcesses	An entity represents a workflow process that defines how a request approval is handled, storing the workflow name and description

19	WorkflowRequests	An enttiy represents a request following a workflow process, linked to a specific workflow process, storing the current request status and related information
20	WorkflowRequestOnTrackHistories	An entity represents the tracking history of a workflow request, linked to a specific workflow request, storing past statuses and related information
21	Transitions	An entity represents a state transition within a workflow process, defining how a workflow request moves from one status to another
22	ActionTriggers	An entity represents an action trigger within a workflow process, defining conditions to automatically trigger transitions or actions based on the user's role, with the associated transition specifying how the status changes after the action is triggered
23	ProcessUsers	An entity represents a group of user roles required to satisfy conditions for triggering an action trigger.
24	ProcessStates	An entity represents a state in a workflow process, storing the state name and related information.
25	UserGroups	An entity represents a group definition based on user roles, storing the group name and the required roles needed to form the group
26	SupportTickets	An entity represents a support ticket created by a user, storing the ticket request details, current status, and the resolution message after the issue is processed
27	Coins	An entity represents a coin balance owned by a user, storing the owner information and the total available coin balance
28	TransactionCoins	An entity represents a coin transaction, storing the sender, the amount of coins transferred, and the receiving user
29	Transactions	An entity represents a real money transaction, storing the sender, recipient, transaction amount, and the transaction status
30	SystemWallets	An entity represents a coin balance owned by a platform, storing the platform information and the total available coin balance
31	WithdrawalRequests	An entity represents a request to withdraw coins as real money, storing the requesting user, the amount of coins to be converted, and the user's bank transfer information
32	RevenueShares	An entity represents the current revenue share between the platform and developers or designers, storing the percentage of the product's purchase price that the platform retains as profit

33	Roles	An entity represents a role in the system, defining the responsibilities and access level assigned to a user within the platform.
34	UserRoles	An entity represents the association between users and roles, storing the specific role assigned to each user in the system.
35	RolePermissions	An entity represents the permissions associated with a role, storing the specific actions or resources that the role can access or modify.
36	Permissions	An entity represents a specific permission or action that can be assigned to a role, defining what users in that role are authorized to do within the system.
37	ApiPermissions	An entity represents the API-related permissions, storing the specific API endpoints or actions that a user role is allowed to access or perform.
38	Game Package	An entity represents a game package, associated to a gameproduct entity, storing details about the game package's content, price, and associated information for distribution.
39	Game Product	An entity represents a product that stores common information shared between game packages or asset packages, including details like name, description, and price
40	Notification System	An entity represents a system for managing notifications, storing information about notification types, recipients, and status for each notification event.
41	Review Request	An entity represents a request for review, storing the item or content to be reviewed, the requester's details, and the status of the review, and is associated with a WorkflowRequests entity to track the current request status
42	System Category	An entity represents a category within the system, storing information related to the categorization of products, assets, or other system elements for organizational purposes.
43	Game Item	An entity represents an individual game item, linked to the Project entity that provides the API key, storing details about the item, including its price, attributes, and association with the project.
44	Asset Package	An entity represents a package of assets associated with a GameProduct entity, storing details about a group of related assets, including price, content, and distribution-related information
45	Feedback Package	An entity represents a user feedback submission related to a specific game or asset package, storing the rating, comments, and user details for evaluating the quality or user experience of the package

Table 17 - Entity Description

3.2 Authentication & Authorization

3.2.1 Sign Up

Actor(s): Guest

- Function trigger: The user presses the "Register" button on the welcome screen.

- Function description: Allows users to create a new account to access the platform.
- Function details:
    - This function requires the user to enter a Username, Email, Password, and Confirm Password.
    - Password must meet security requirements: at least 8 characters, one uppercase letter, one number, and one special character.
    - The "Create account" button will be enabled only when all fields are filled correctly.
    - After successful registration, the user is redirected to the login or home screen.
    - There is an option to navigate to the Sign In page if the user already has an account.

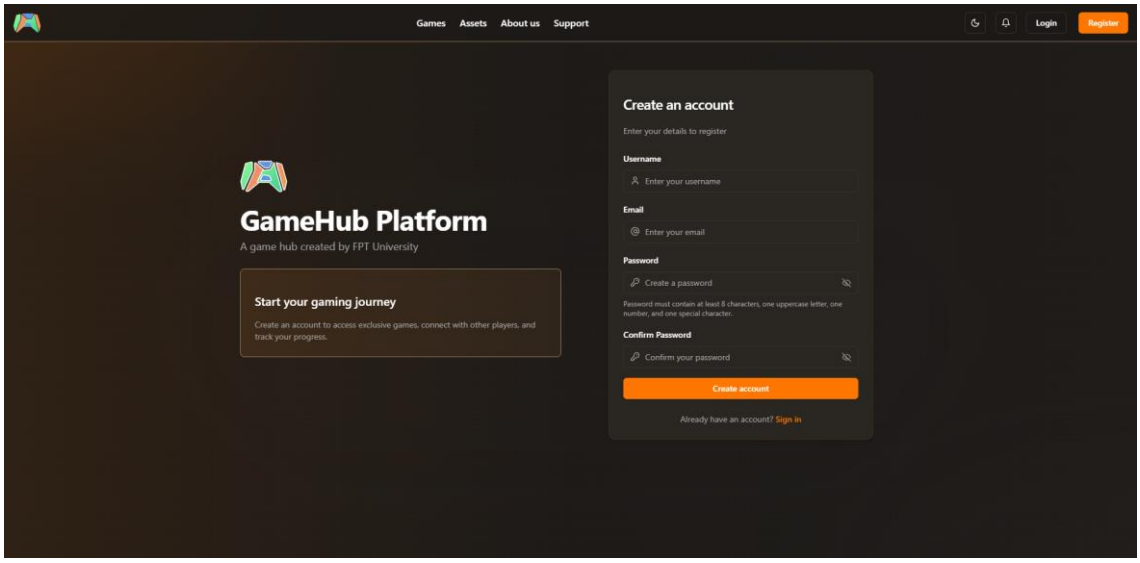

Figure 10 - Customer Register Account

3.2.2 Login

Actor(s): Customer, Developer, Designer, Admin, Moderator

- Function trigger: A user submits their login credentials via the sign-in form.
- Function description: This feature allows users to authenticate into the system by providing their username and password. Based on the authenticated user's role, the system navigates them to the appropriate landing page: customers go to the homepage, while developers, moderators, designers, and admins are directed to the management dashboard.
- This function authenticates the user's login credentials (username and password) to authorize access to system resources.
    - Abnormal cases:
        - Invalid credentials: API returns 401 Unauthorized. Show error message: “Invalid username or password.”
        - Missing fields: Frontend should validate and prompt: “Please enter both username and password.”
        - Account locked or disabled: API returns appropriate error; display: “Your account is disabled. Please contact support.”
    - Success Case:
        - The user is authenticated and redirected to the appropriate screen based on their role. Session tokens or cookies are securely stored for future authenticated requests.
    - Precondition:
        - Users must have a valid account in the system.

        - Backend authentication service must be available.
    - Postcondition:
        - The user is authenticated and navigated to the correct interface.
        - Token/session data is stored securely on the client side.

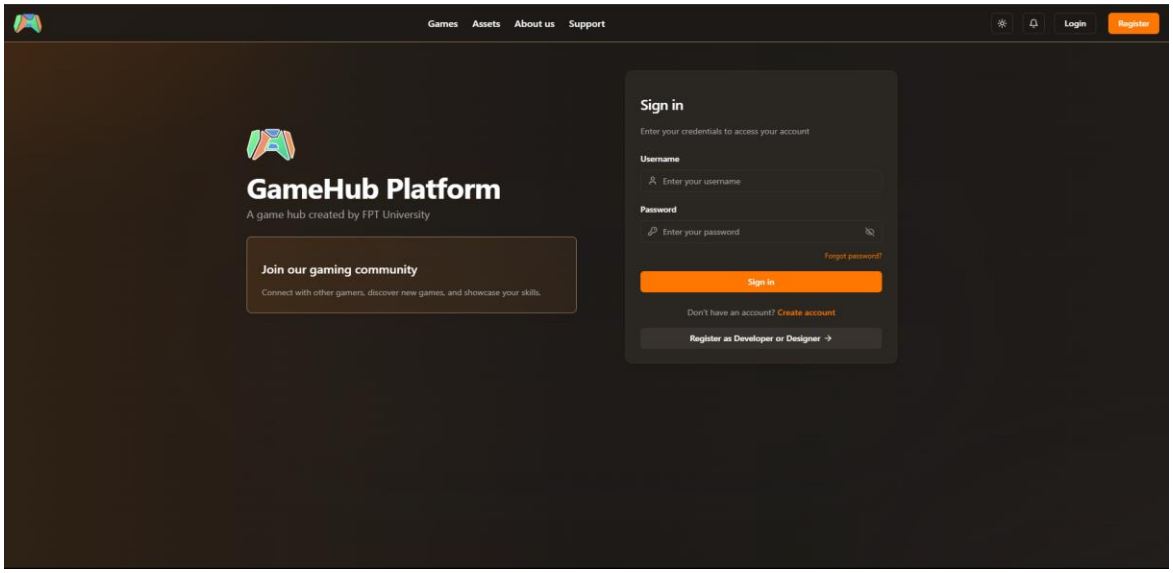

Figure 11 - Login

3.2.3 Reset Password

Actor(s): Customer, Developer, Designer

- Function trigger: A user clicks the "Forgot Password" link and submits their email address.
- Function description: This feature allows users who have forgotten their password to initiate a password reset process. The system sends a password reset email to the provided email address if it is associated with an existing account.
- Function details: The front-end application sends a POST request to /api/auth/Login with the username and password.
    - Abnormal cases:
        - Invalid or non-existent email: API may return 400 Bad Request or 404 Not Found. The frontend should show a generic message:
        - “If the email exists in our system, a reset link has been sent.” (This avoids disclosing account existence.)
        - Email sending failure: Retry mechanism or display error: “Failed to send reset email. Please try again later.”
        - Missing email field: Client-side validation should prevent submission without a valid email.
    - Success Case:
        - If the email is valid and exists, the user receives an email with a reset password link.
    - Precondition:
        - The user has an existing account with a valid email.
        - Mail service is properly configured and operational.
    - Postcondition:
        - An email with a reset password link is sent to user.

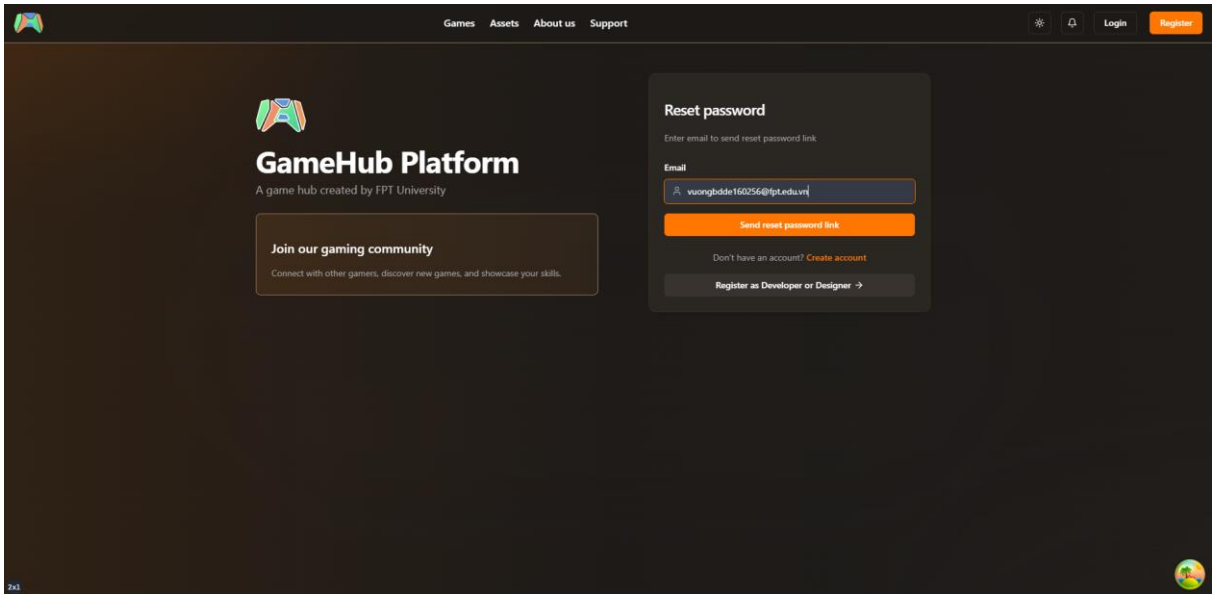

Figure 12 - Reset Password

3.2.4 Register Developer/Designer Account

Actor(s): Guest

- Function trigger: User uploads a student ID card via the registration page for developers/designers.
- Function description: This feature allows eligible students to register for a Developer or Designer account by verifying their student ID. The system uses AI to process the uploaded card and cross-checks with the student database before sending a registration link.
- Function detail:
    - This function requires user following these steps:
        1. The user visits select the option register developer account
        2. The user is required to upload a valid student ID card.
        3. The system asynchronously triggers an AI-based validation application that:
        4. Extracts the student ID from the uploaded image.
        5. Compare it with the database of valid students.
        6. If a match is found, retrieve the associated email address from the database.
        7. Send a registration link to that email.
        8. When the user clicks the link, they are redirected to a signup page with a form for registering the developer account.
    - Abnormal cases:
        - Invalid or non-existent email: API may return 400 Bad Requests or 404 Not Found. The frontend should show a generic message:
        - “If the email exists in our system, a reset link has been sent.” (This avoids disclosing account existence.)
        - Email sending failure: Retry mechanism or display error: “Failed to send reset email. Please try again later.”
        - Missing email field: Client-side validation should prevent submission without a valid email.
    - Success Case:
        - If the email is valid and exists, the user receives an email with a reset password link.
    - Precondition:
        - The user has an existing FPT student ID with a valid email.

        - Mail service is properly configured and operational.
    - Postcondition:
        - A Developer/Designer account that is associated with FPT student ID is created.

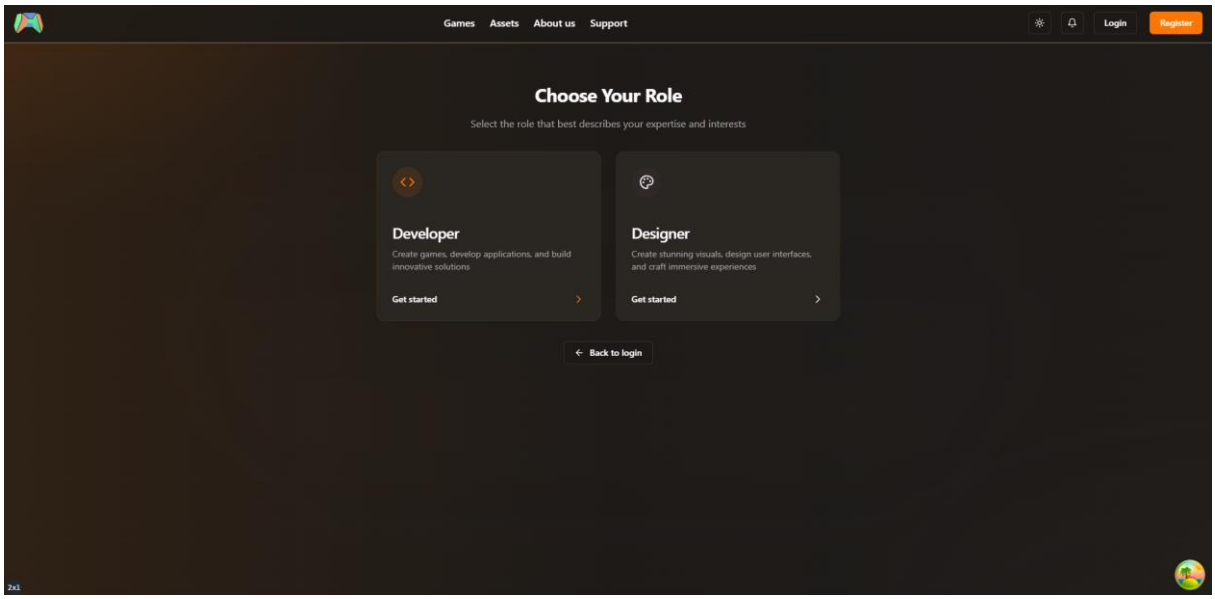

Figure 13 - Register Developer/Designer Account

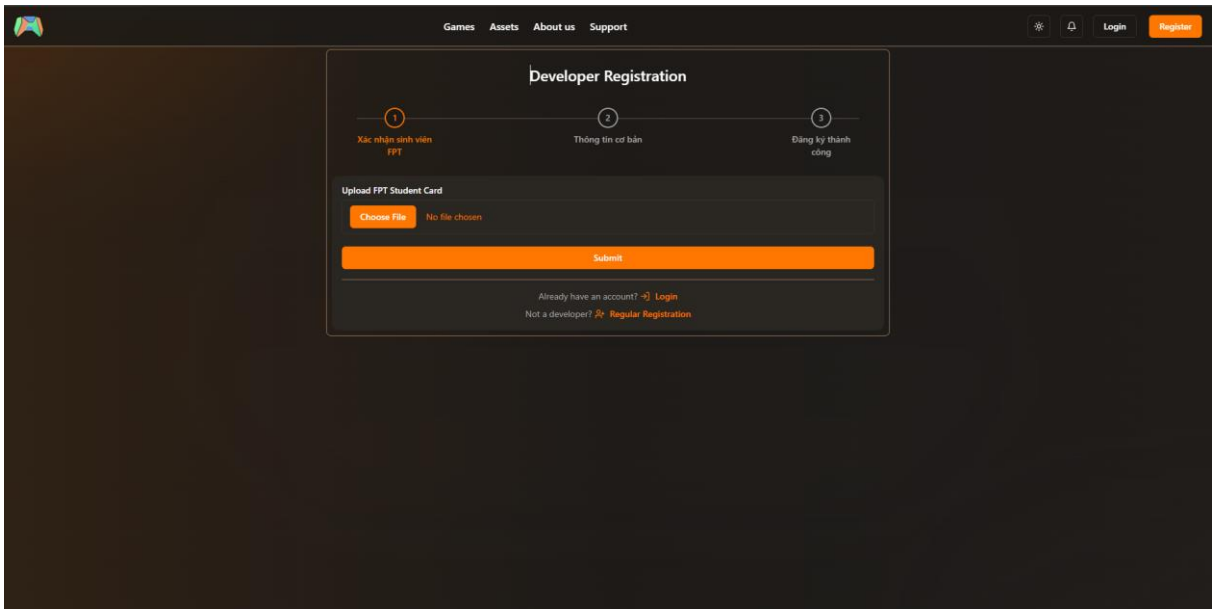

Figure 14 - Form Register Developer Account

3.3 Notification Management

3.3.1 View History Of Notification

Actor(s): Customer, Developer, Designer, Admin, Moderator

- Function trigger: The user clicks on the bell icon in the header bar or navigates to the “Notifications” section.
- Function description: This function displays a paginated list of past notifications, including read and unread ones. Notifications are ordered by most recent first and grouped by date.
- Function details: This function return a list of notifications are sorted in descending order with the following fields:

- Icon (based on type: comment, approval, etc.)
- Message (short text, e.g., "Admin approved your request")
- Timestamp
- Read/Unread status (bold for unread)

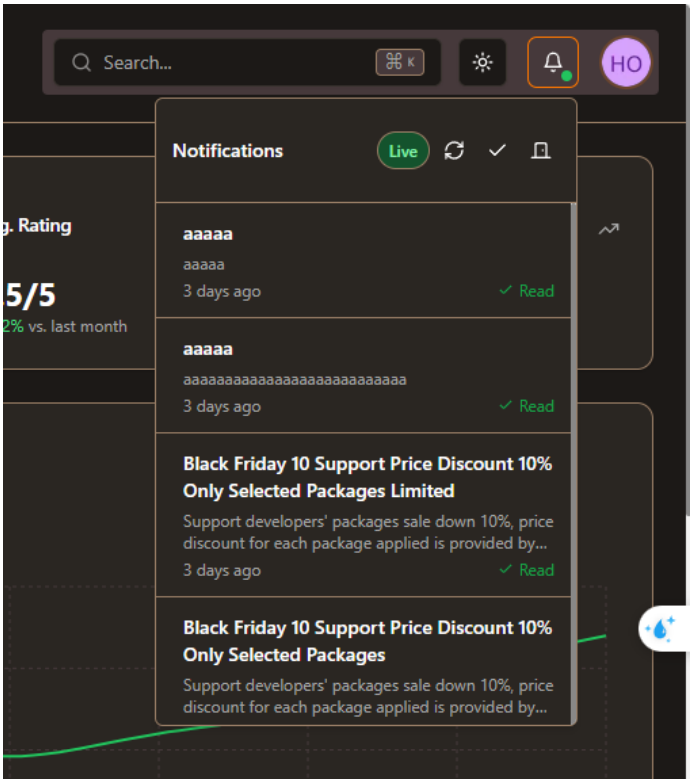

Figure 15 - View Notification

3.3.2 Read Notification In Details

Actor(s): Customer, Developer, Designer, Admin, Moderator

- Function trigger: User clicks on a notification item from the list or bell dropdown.
- Function description: Upon clicking a notification, the system redirects the user to the relevant screen or record related to the notification, and marks it as read.
- Function details:
    - If the notification refers to a comment, the user is redirected to that comment thread.
    - If the notification refers to an approval, the system opens the detail screen of that item.
    - Business Rules:
        - Once a notification is opened, it is automatically marked as "read".
        - System logs when the user reads the notification.

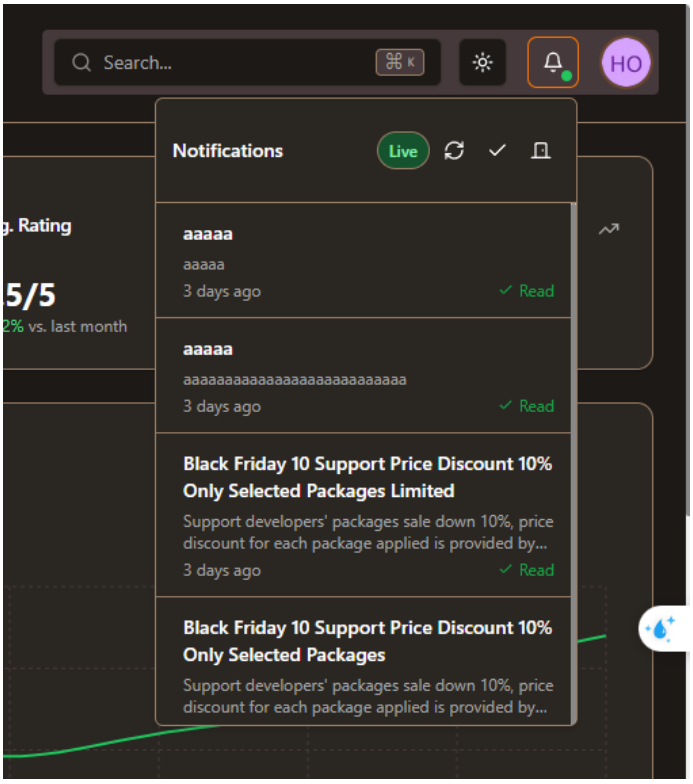

Figure 16 - Read Notification

3.4 System Role Management

3.4.1 Add New System Role

Actor(s): Admin

- Function trigger: The Admin accesses the "System Roles" screen from the system administration module and clicks on the "Add New" button located at the top-right of the roles list table.
- Function description: This function allows the Admin to create a new system role, which is used to manage access permissions for internal users. A role can be assigned to multiple users.
- Function details:
    - This function require admin inputs the following information:
        - Role Name: role name is required and must be unique in the system
        - Description (optional)
    - Business Rules:
        - A role can be assigned to multiple users.
        - A user has more than one role.
    - Abnormal cases:
        - If the role name is missing, the system displays: “Role name is required.”
        - If the role name already exists, the system displays: “Role name already exists.”
    - Success-case:
        - The system redirects back to the roles list and displays a success message: “Created role successfully.”

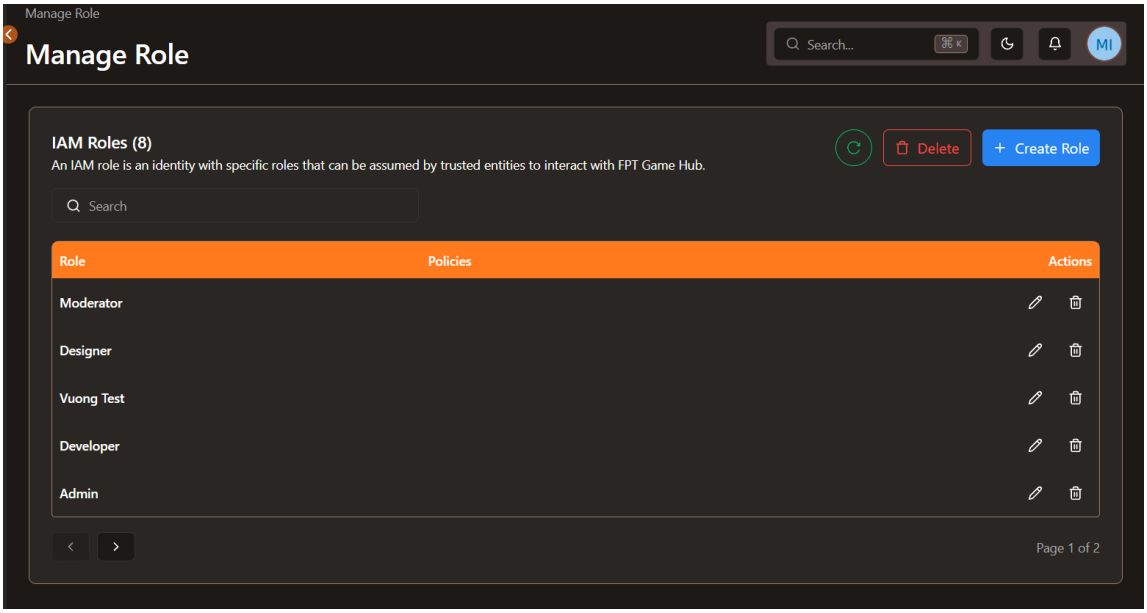

Figure 17 - Roles List

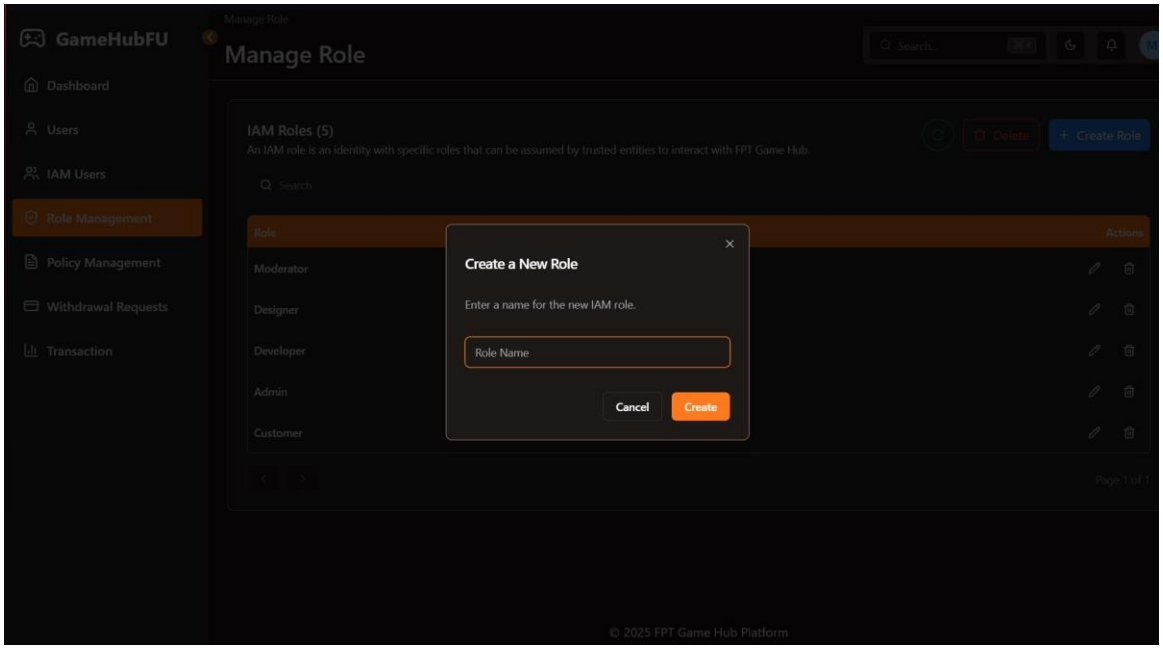

Figure 18 - Create New Role

3.4.2 View All System Roles In List

Actor(s): Admin

- Function trigger: The Admin navigates to the "IAM Roles" screen via the system administration menu.
- Function description: This function allows the Admin to view a complete list of all system roles created in the application. The list includes role names, descriptions and available actions (e.g., Edit, Delete).
- Function details: This function returns a list of roles in the system. Admin can view all system roles that exist in the system to manage system roles in the table which is have Role, Policies and Actions (Edit or Delete)
    - Validation Rules: Ensure data is correctly retrieved and displayed from the system.

- Business Rules:
  - Admin can view all roles regardless of who created them.
- Abnormal cases:
  - If the role list is empty, display a message: “No roles found.”
  - If there is a system error retrieving data, show: “Unable to load roles. Please try again later.”

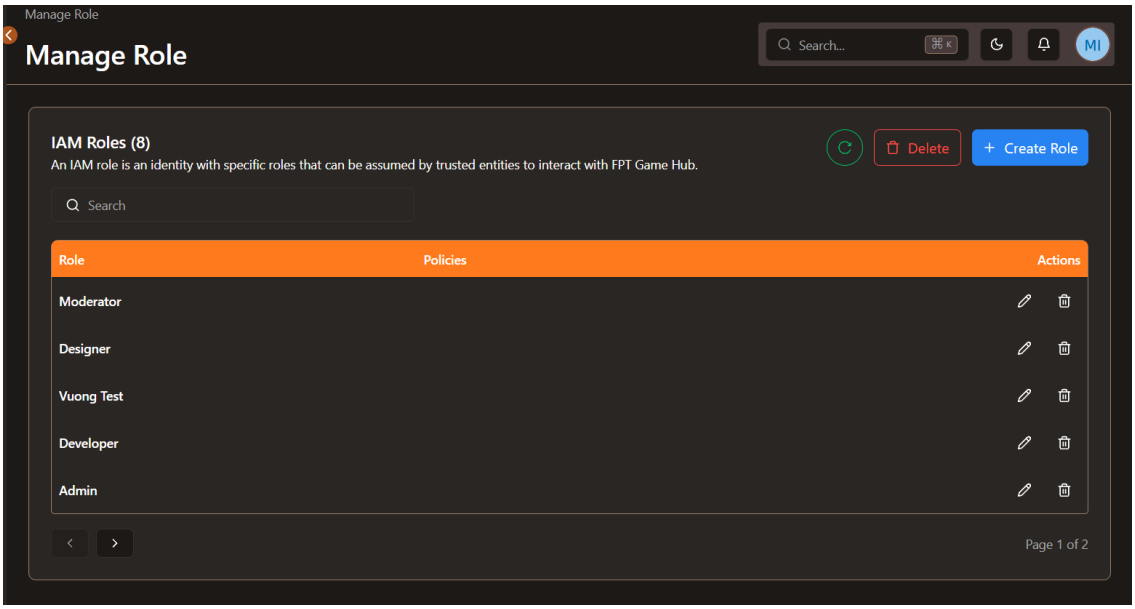

Figure 19 - Manage Roles List

3.4.3 Assign Role To System Account

Actor(s): Admin

- Function trigger: The Admin accesses the “User Management” or “System Accounts” screen, selects a specific user account, and clicks the "Assign Role" button or tab.
- Function description: This function allows the Admin to assign one or more roles to a system user account. The assigned roles determine what actions the user can perform in the system.
- Function details: This function displays available system roles in a table for the Admin to select and assign to the chosen user account, then save the changes by clicking the "Save" button.
- Validation Rules:
  - At least one role must be selected.
  - Only valid system roles can be assigned.
  - Business Rules:
    - Each account can have more than one role.
    - Changes take effect immediately or after the user’s next login, depending on implementation.

- Abnormal cases:
  - If no roles are selected and the user tries to save, show: “Please select at least one role.”
  - If the system fails to update the assignment, show: “Failed to assign roles. Please try again.”

3.5 System Role Permissions Management

3.5.1 Add New Policy Permission

Actor(s): Admin

- Function trigger: The Admin navigates to the "Manage Policy" screen and clicks the “Create Policy” button located at the top-right corner of the screen.
- Function description: This function allows the Admin to create a new IAM permission (policy) that defines a specific access right or system capability. These permissions can later be attached to roles.
- Function details: This function shows the admin the form to create new system policy, admin input the new policy name, click the button “Create” for save changes.
    - Validation Rules: The policy name must:
        - Not be empty.
        - Be unique across all existing policies.
    - Abnormal cases:
        - If the input field is empty or contains invalid characters, the system displays: “Policy name is required and must follow naming rules.”
        - If the name already exists, the system displays: “A policy with this name already exists.”
        - If a system error occurs during creation, the system displays:“Failed to create policy. Please try again.”
    - Success case:
        - The modal closes automatically and a confirmation toast is shown:

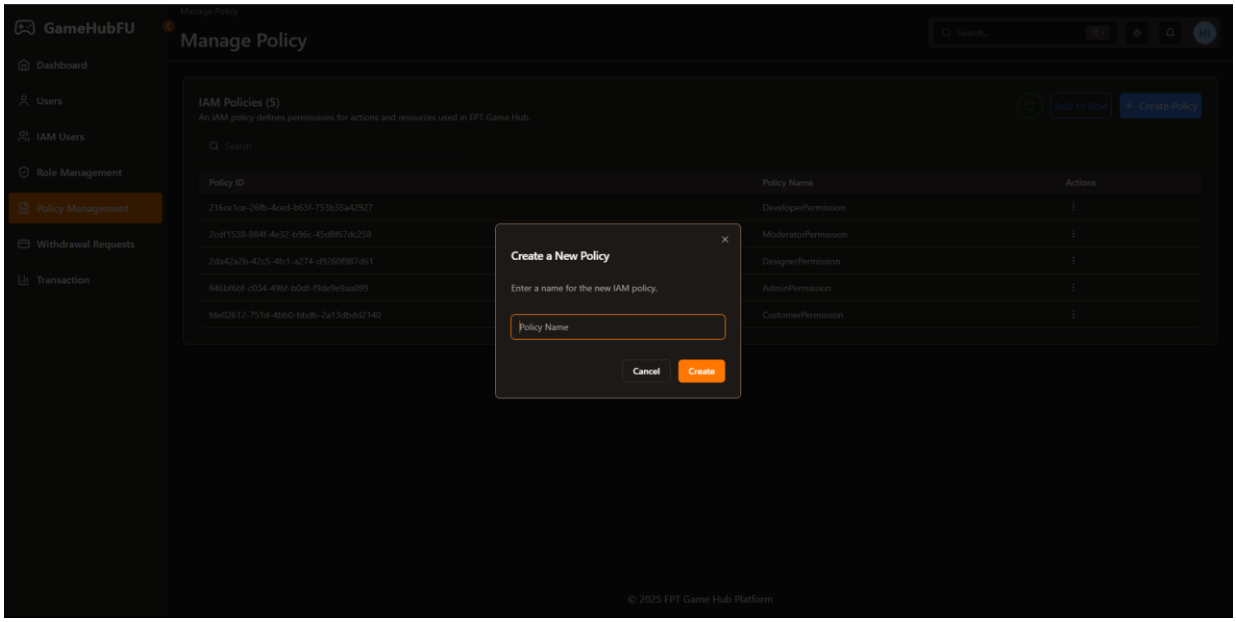

Figure 20 - Create New Policy

3.5.2 View All Permissions In List

Actor(s): Admin

- Function trigger: The Admin navigates to the "Manage Policy" screen via the system menu or dashboard.
- Function description: This feature enables Admins to view a complete list of all existing IAM policies (permissions) that define access rights within the FPT Game Hub platform.
- Function details: The function displays all IAM policies (Permission).
    - The Admin can do following actions:

        - Admins can scroll through the list, use the search box to filter, or interact with individual policy entries.
        - Sorting or pagination may be applied if the number of entries is large.
    - Business Rules:
        - Only users with Admin role or equivalent permissions can access this page.
        - Each policy must be shown with accurate metadata (ID, name).
    - Abnormal cases:
        - If no policies exist, the system shows a message: “No IAM policies found.”
        - If the system fails to fetch data, an error appears: “Unable to load policies. Please try again.”
    - Success case:
        - All available IAM policies are displayed correctly in the table.

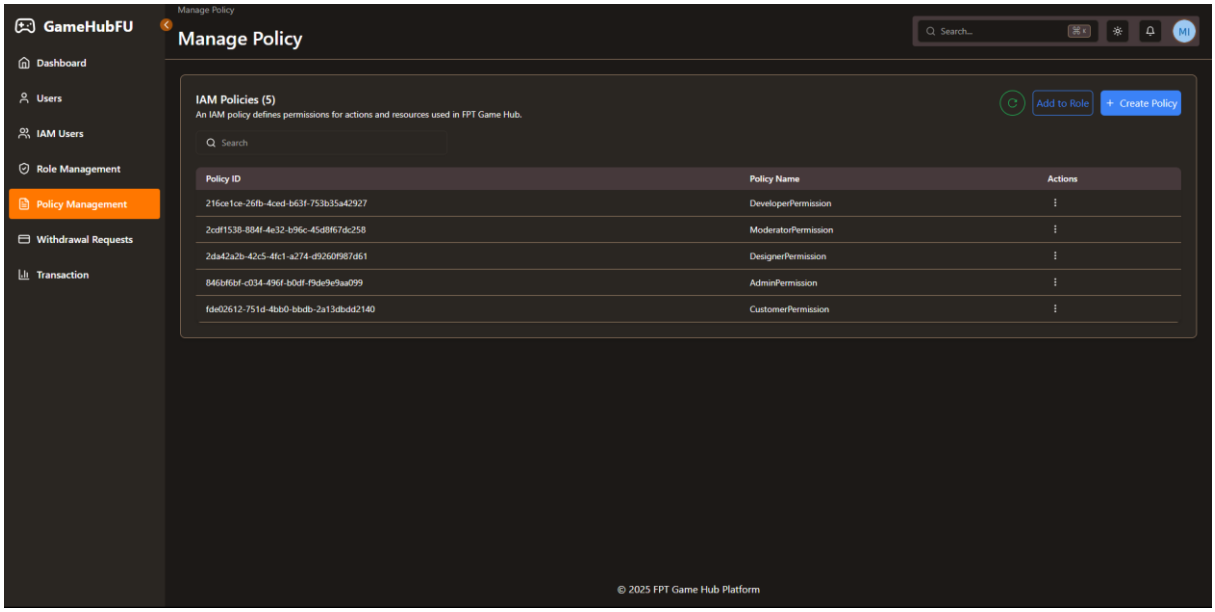

Figure 21 - Manage Policy

3.5.3 Assign Policy Permissions to Role

Actor(s): Admin

- Function trigger: The Admin accesses the "Manage Role" screen, selects an existing role from the list, and clicks the "Edit" icon button next to the role.
- Function description: This function allows the Admin to edit the name of an existing system role and manage its attached policies (permissions). The Admin can add new policies or remove existing ones from the role.
- Function details: This function displays the current role name in a text box and its permissions as a checkbox list. The Admin can rename the role, modify its permissions, and click "Save" to apply the changes.
    - Validation Rules:
        - Role’s name must not be empty and must be unique across the system.
    - Business Rules:
        - Certain system-level policies may be restricted from being attached to standard roles.
        - Policy changes take effect immediately for users assigned to the role.
    - Abnormal cases:

- If the role name is left empty or conflicts with another role, show: “Role name is required and must be unique.”
- If saving fails due to a server error, show: “Unable to save changes. Please try again.”

- Success case:
  - On successful update, the modal closes and a confirmation message is shown:“Role updated successfully.”

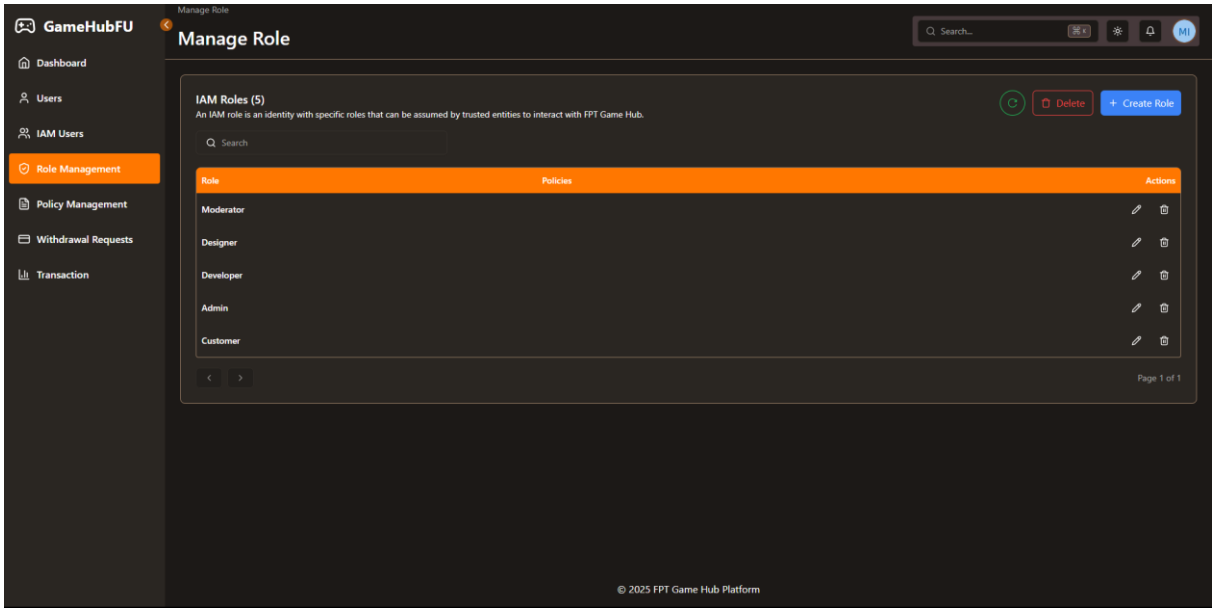

Figure 22 - Select Role From Roles List

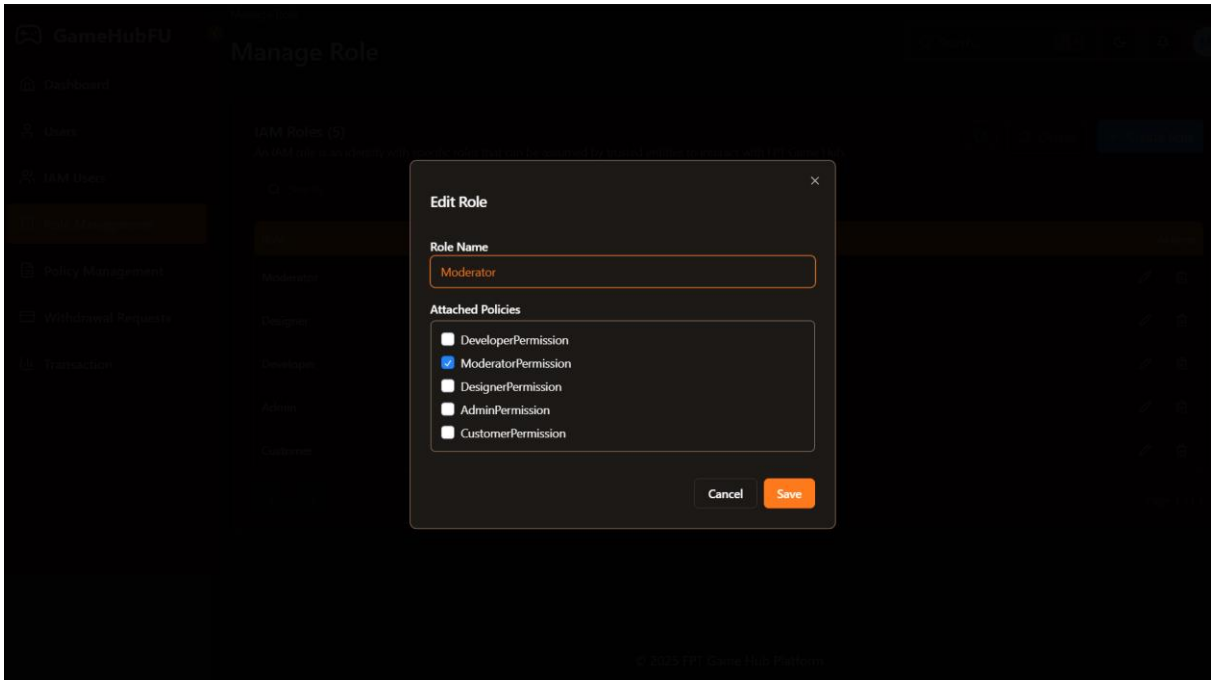

Figure 23 - Assign Role Policy

3.5.4 Update Policy Permissions

Actor(s): Admin

- Function trigger: The Admin navigates to the "Manage Policy" screen via the system menu or dashboard.
- Function description: This feature allows Admins to update policy permission.
- Function details: This function displays the current policy permissions and a list of API paths as multi-select checkboxes. The Admin can update the policy by selecting or deselecting API paths assigned to the permission.
    - Business Rules:
        - Only Admins or users with equivalent permissions are allowed to api paths from current policy permission.
    - Abnormal cases:
        - If no API paths are available for selection: *"No API permissions found."*
        - If the system fails to remove permissions: *"Failed to update policy permission. Please try again."*
    - Success case:
        - When API path permissions are successfully updated for the selected policy, the system displays: *"Policy permission updated successfully."*

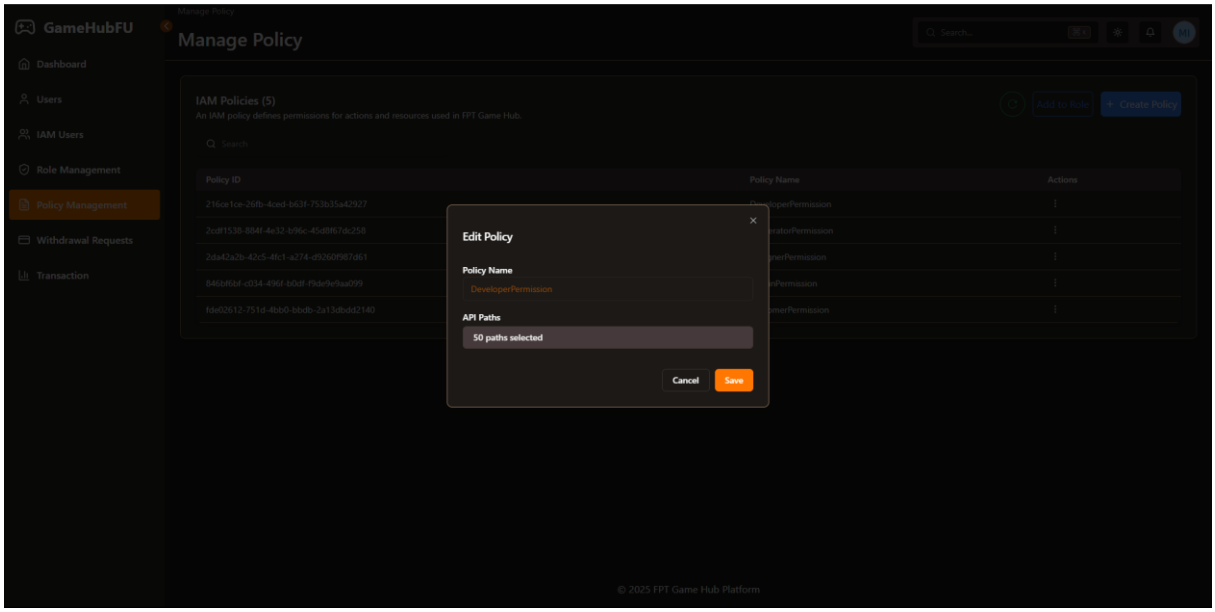

Figure 24 - Edit Policy

3.6 API Document Management

3.6.1 Add New API Document

Actor(s): Moderator

- Function trigger: Moderator accesses the "API Documents" section in the system administration module and clicks the "+" icon button.
- Function description: Allows Moderator to create a new API documentation using .md files. Moderators need to input file path of .md files
- Function details: This function provides a text editor for the Admin to create and publish new API documentation, making it accessible to developers for reading integration instructions.
    - Success Case:
        - Successfully add a new API document
    - Abnormal Cases:

- User lacks permission: “403 Forbidden”
- Network interruption: Show error or retry prompt depending on the implementation.

- Postcondition:
  - A new API document is created and visible in the list for reference and management.

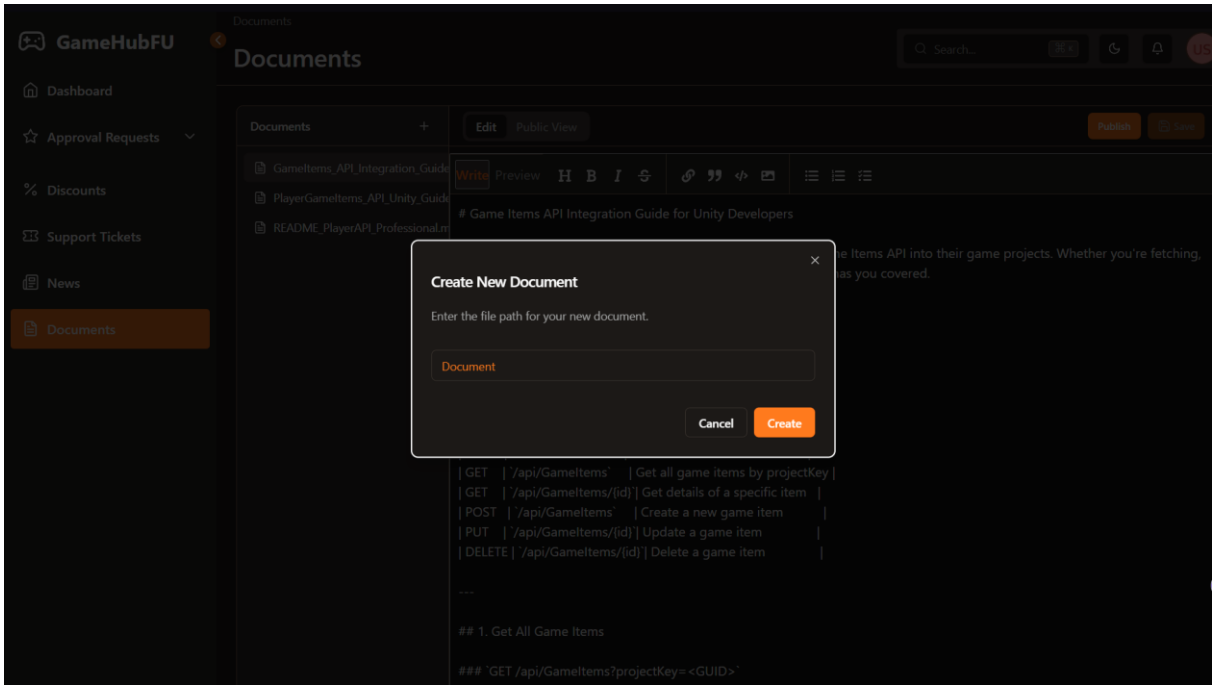

Figure 25 - Create Document

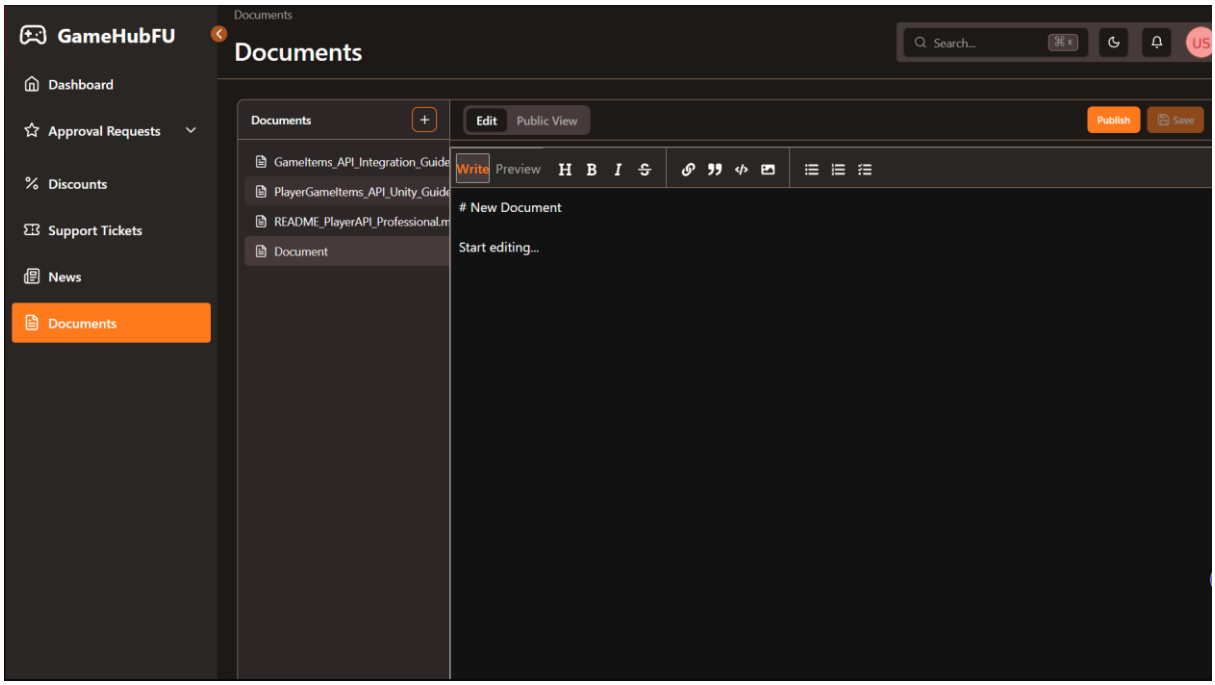

Figure 26 - Write Document

3.6.2 Update API Document

Actor(s): Moderator

- Function trigger: The Moderator navigates to the "Documents" section via the system menu (icon with document symbol).
- Function description: This feature enables Moderators to update the content of existing API documents, including integration guides and endpoint specifications, and publish the changes to make them available to developers and other authorized users.
- Function details: This function allows developers to view documentation for all available APIs to support integration with their games. Moderators can also access the full list of API documents to review and manage them within the system.
    - Abnormal Cases:
        - No documents available: Display a message such as “No documents found” or show an empty state UI.
        - User not authenticated or lacks permission: Redirect to the login screen or show “403 Forbidden.”
    - Success Case:
        - Moderator successfully updates and publishes the API document.
        - Developers and other authorized users can view or copy the updated document.
    - Postcondition:
        - The updated API document is saved and published, and it becomes accessible to authorized users for reference and integration.

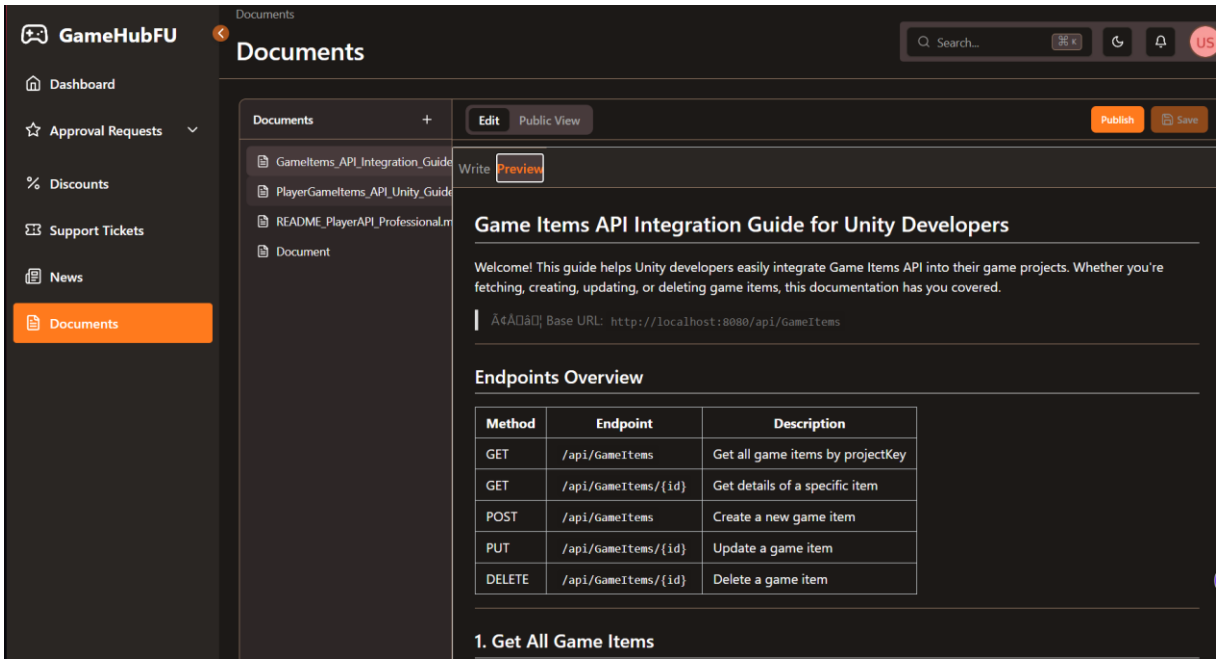

Figure 27 - Update Document

3.6.3 Read API Document

Actor(s): Developer

- Function trigger: The Developer selects a document from the "Documents" list to view its full content.
- Function description: This feature allows Developers to read the detailed content of any API document available in the system. The document displays integration guidelines, API endpoints, parameters, expected responses, and usage instructions.

- Function details: This function allows Admins and Developers to view the full content of an API document from the "Documents" list. Each document includes integration guidelines, API endpoints, parameters, response formats, and usage instructions. Moderators can additionally view public sharing settings and copy the public URL for distribution.
    - Abnormal Cases
        - Document not found: “Document not found or no longer available.”
        - Insufficient permissions: “403 Forbidden” and back to login
    - Success Case
        - Successfully view an API document

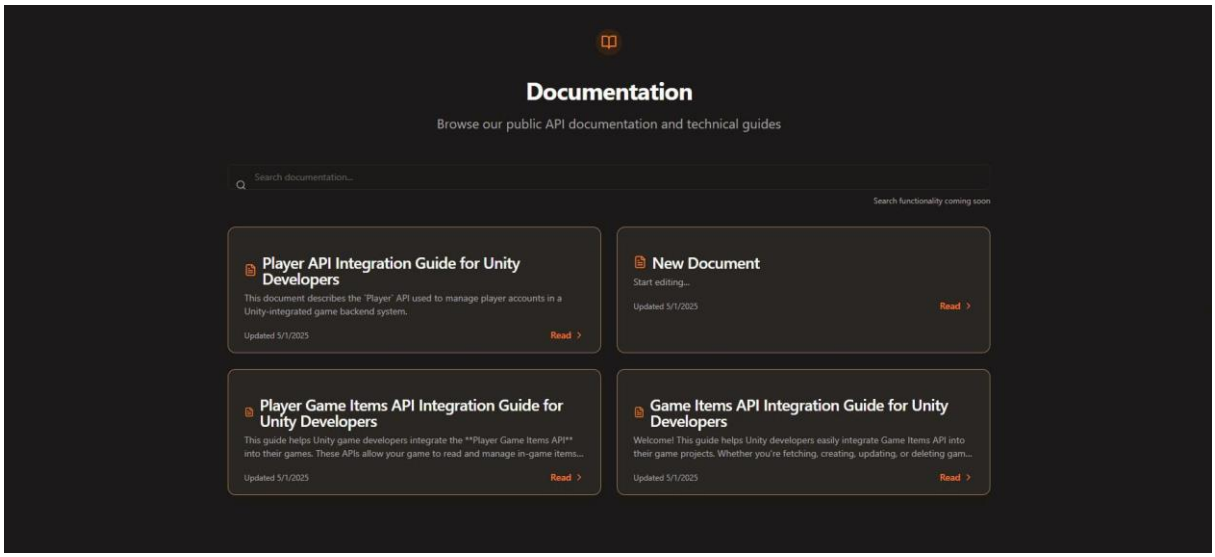

Figure 28 - Read Published Documents

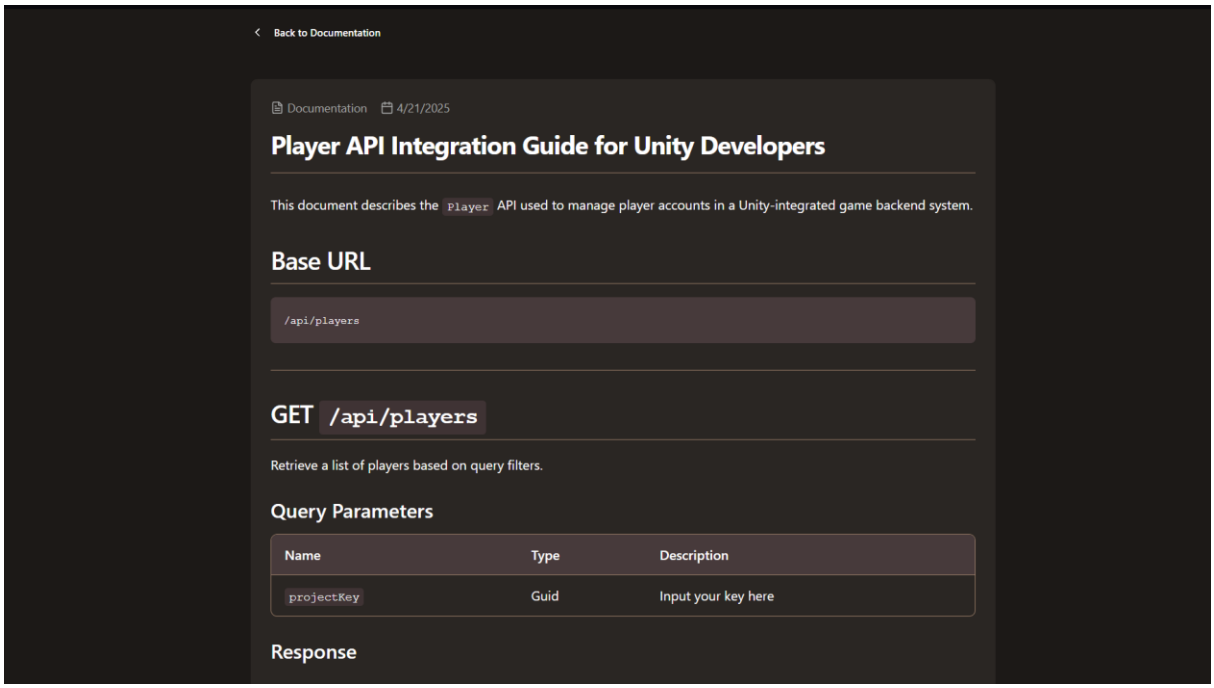

Figure 29 - Read Document In Details

3.7 Project APIs Integration Management

3.7.1 Add New Project

Actor(s): Developer

- Function trigger: The Developer clicks the "Create Project" button located at the top-right corner of the "Projects" screen.
- Function description: This feature allows Developers to create a new project by entering the project name. Projects act as containers for managing APIs integration.
- Function details: This function allows the Developer to create a new project by entering the project name. Once submitted, the system validates the input and creates a new project record associated with the Developer. The newly created project is then added to the list and used as a container to manage APIs integration.
    - Abnormal Case:
        - Project name is empty: “Project name is required.”
        - Server/database failure: “Failed to create a project. Please try again later.”
    - Success Case:
        - Create a new project successfully
    - Precondition:
        - Developer authorized
        - Developer already in the project screen.
    - Postcondition:
        - New project created with developer id following.

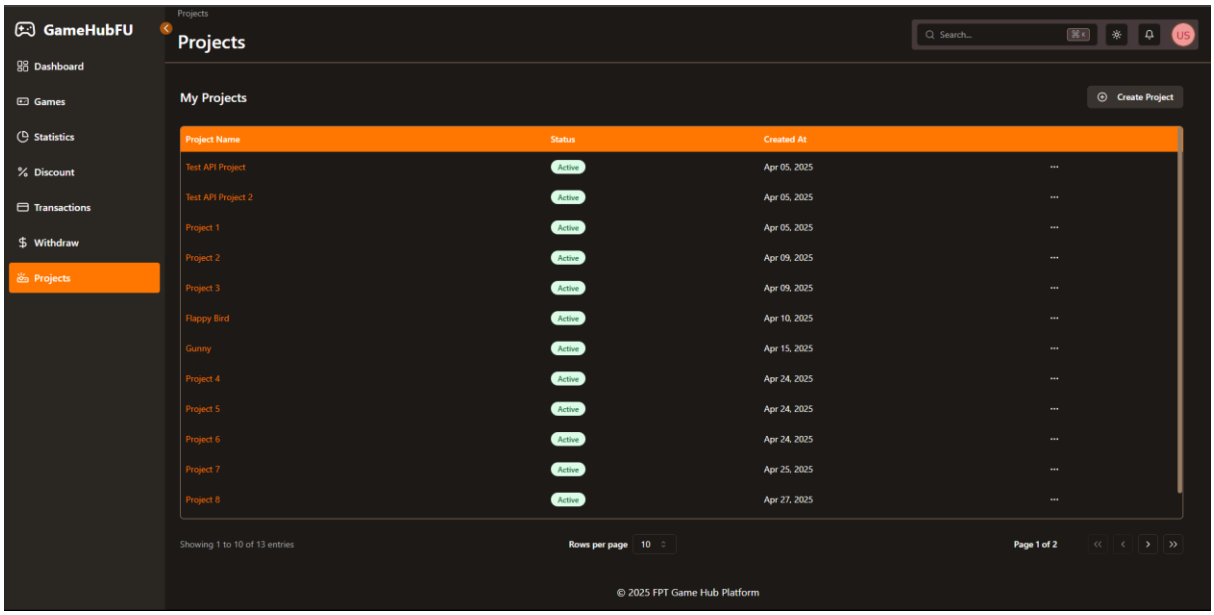

Figure 30 - Get Current Game Projects

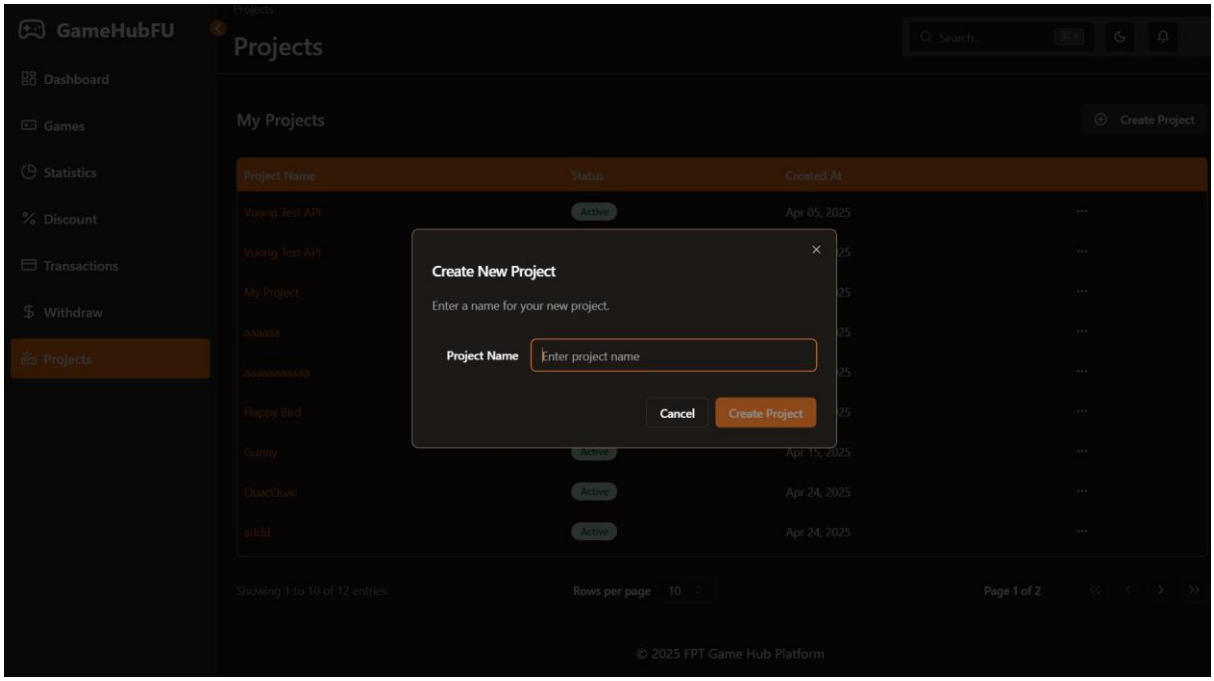

Figure 31 - Create New Game Project

3.7.2 View All Project In List

Actor(s): Developer

- Function trigger: The Developer navigates to the "Projects" section from the sidebar menu.
- Function description: This feature allows Developers to view a paginated list of all projects they have created. Each project includes essential information such as project name, status, and creation date for easy management.
- Function details: This function retrieves and displays a paginated list of projects associated with the Developer. Each item in the list shows the project's name, status, and creation date. The system loads up to 10 records per page by default and allows navigation between pages. The data is sorted by creation date in ascending order. The function enables developers to easily monitor and manage their created projects from a centralized interface.
    - Abnormal Case:
        - Developer has no projects: "Nothing to display. You haven't created any project yet."
        - Server/database failure: "Failed to load project list. Please try again later."
        - Network error during fetch: "Unable to connect. Please check your network and try again."
    - Success Case:
        - Developer successfully views their project list.
    - Precondition:
        - Developer is authorized and logged in.
        - Developer accesses the project list from the sidebar.
    - Postcondition:
        - A list of projects created by the Developer is displayed, along with pagination.

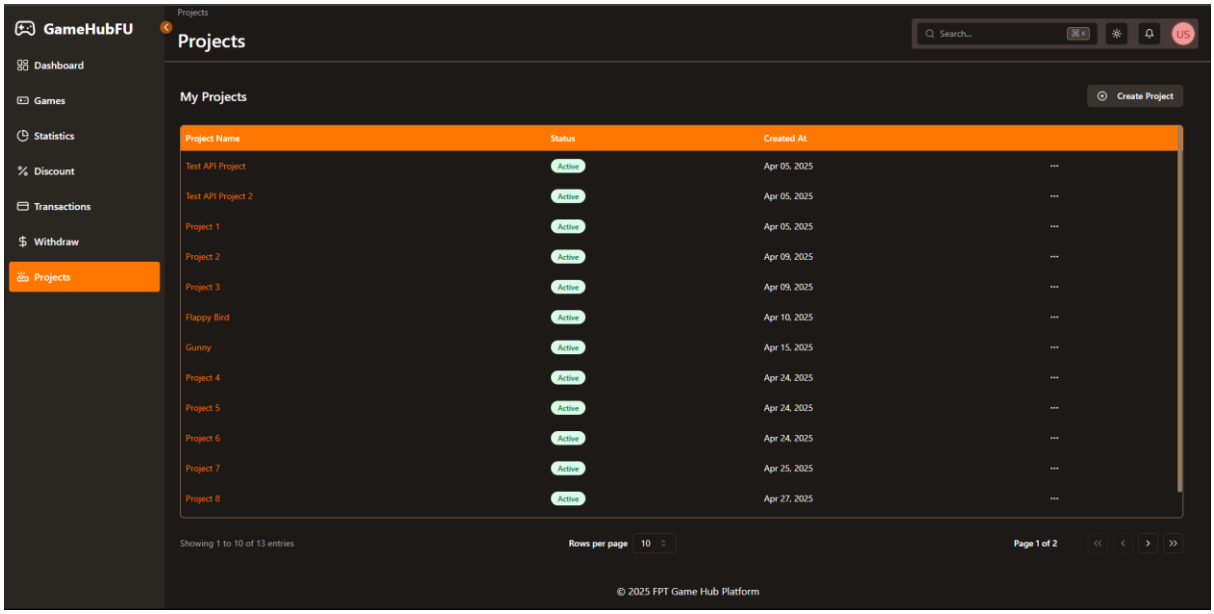

Figure 32- Projects List

3.7.3 View Project Information

Actor(s): Developer

- Function trigger: The Developer selects a project from the "Projects" list to view detailed information.
- Function description: This feature allows Developers to view comprehensive information about a selected project. The screen provides a detailed breakdown across three tabs: Overview, Players, and Game Items, facilitating easy monitoring and management of project-related data.
- Function details: This function displays all relevant information about a selected project. When a Developer selects a project from the list, the system fetches and loads project metadata, player data, and associated game items. By default, the Overview tab is shown first, containing general project information and statistics. Developers can switch to the Players tab to view or manage the list of players, or to the Game Items tab to inspect or manage in-game items tied to the project. The function supports paginated views for large datasets and saves the last active tab to enhance user convenience. Data such as project name, project key, status, player counts, and item rarity is presented in an organized and structured format.
    - Abnormal Case:
        - Project not found or deleted: “The selected project could not be found.”
        - Server/database error when loading details: “Failed to load project details. Please try again later.”
        - Network issue during data fetch: “Unable to load data. Please check your network connection.”
    - Success Case:
        - Developer successfully views the selected project’s detailed information across tabs.
    - Precondition:
        - Developer is authorized and logged in.
        - Developer has at least one project in the list.
    - Postcondition:

- Project details including overview, players, and game items are displayed and interactable.

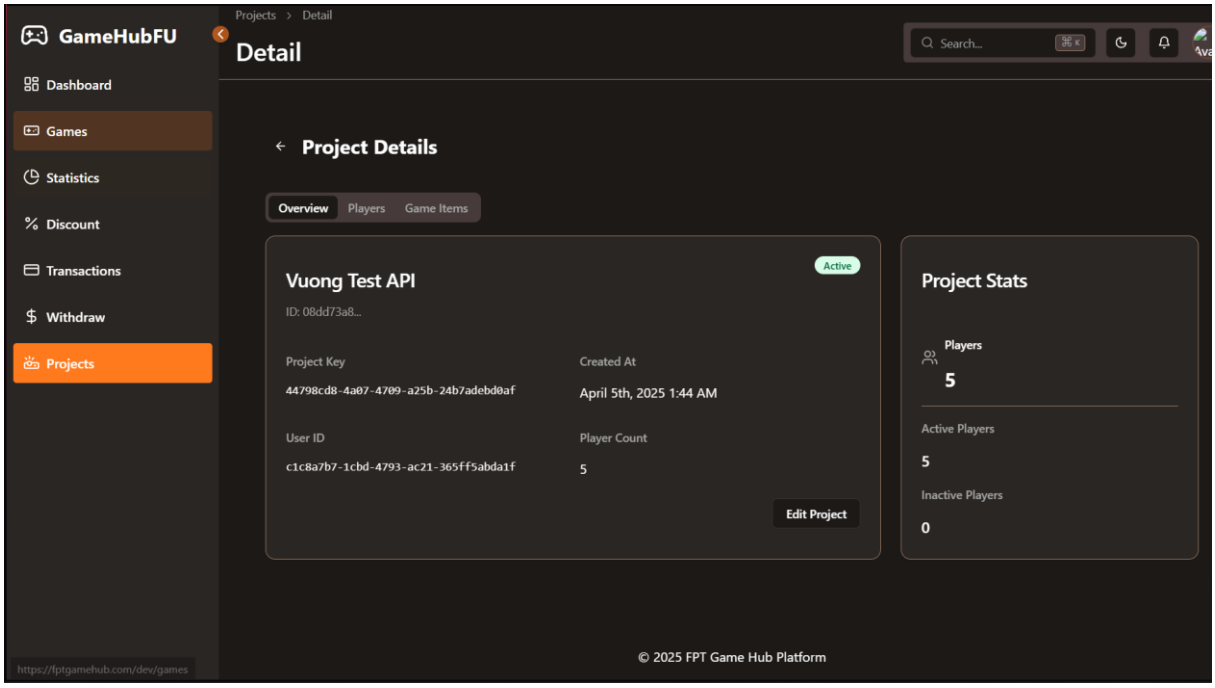

Figure 33 - View Project In Details

3.8 Game Players Of Game Project Management

3.8.1 Integrate API Manage Game Players In Source Game Project

Actor(s): Developer

- Function trigger: After creating a project, the Developer obtains the Project Key displayed in the Project Details screen.
- Function description: This feature enables Developers to integrate player management capabilities into their game's source code by using the provided Project Key. The Project Key acts as a secure identifier linking the in-game API calls with the corresponding project on the platform, allowing for player creation, update, retrieval, and deactivation directly from the game client or server.
- Function details: This function allows the Developer to retrieve the auto-generated Project Key from the Overview tab of the Project Details screen and integrate it into their game's codebase.
    - The system uses this key to authenticate the request, identify the project, and ensure that only authorized actions are performed.
    - All player data created or modified via API is automatically associated with the corresponding project based on the Project Key.
    - The key remains constant unless explicitly reset.
    - Developers are expected to secure this key in their code and environment.
    - Abnormal Case:
        - Missing Project Key in API header: "Project Key is required."
        - Invalid or expired Project Key: "Invalid Project Key. Access denied."
        - Key used in unauthorized context "Unauthorized Project Key usage detected."
        - Key compromised or reused by other parties: "Suspicious activity detected. Contact support to rotate Project Key."
    - Success Case:

- API requests are authenticated using the Project Key, and player-related operations are executed successfully (e.g., player created, updated, or fetched).

- Precondition:
  - Developer has a valid project with a generated Project Key.
  - Developer has access to the Project Details screen.
- Postcondition:
  - The Project Key is used as a secure reference in the Developer's game code.
  - API requests using the key can manage players under the corresponding project context.

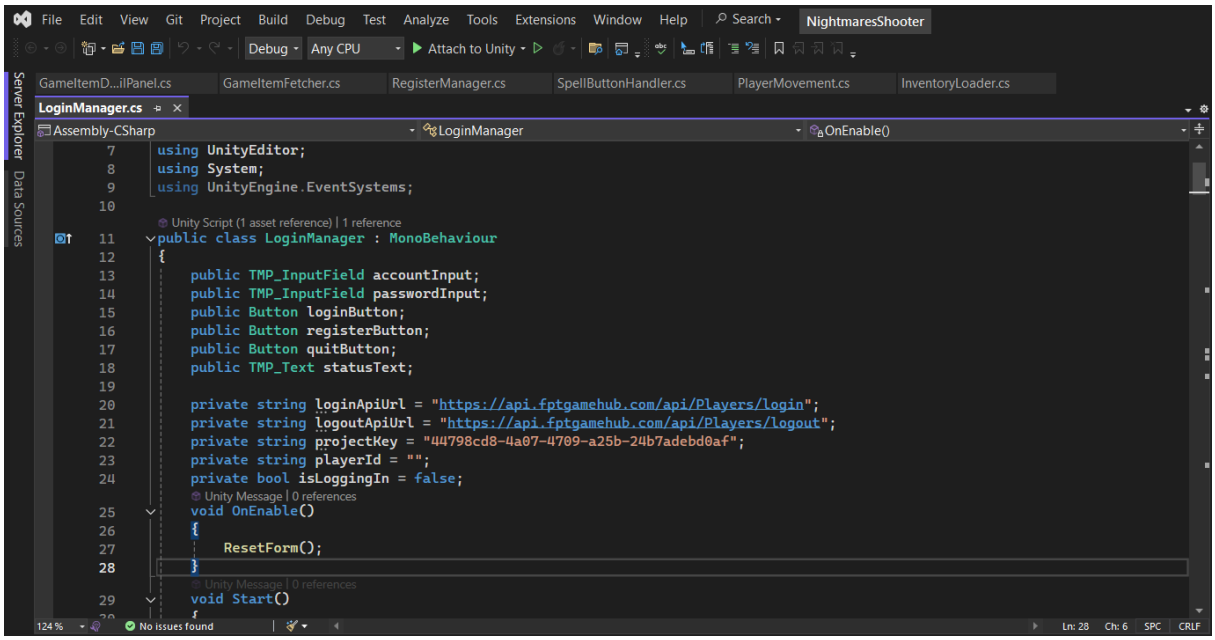

Figure 34 - Embedded Project Key In Developer's Source Game

3.8.2 Monitor Game Players Of Game Project

Actor(s): Developer

- Function trigger: From the Project Details screen, the Developer selects the "Players" tab.
- Function description: This feature allows Developers to view and monitor the list of players who are associated with a specific game project. Developers can track player accounts, display names, total playtime in minutes, and their current active status. This information aids in assessing player engagement and activity levels.
- Function details: This function displays a table of all players linked to the selected project.
  - When the Developer selects the "Players" tab, the system fetches and presents the player list scoped to the current project.
  - Each row contains the player's display name, account, total play time (in minutes), and status (e.g., Active).
  - The player list is refreshed every time the Developer accesses the tab to ensure real-time data.
  - Additional UI elements like the "Add Player" button and placeholder rows for future actions (e.g., edit or deactivate) support extensibility.
  - The displayed total play time is always a non-negative integer, and player statuses are retrieved in real-time from the backend. If the system fails to retrieve data, an error popup is displayed. The function ensures data isolation by only showing players belonging to the selected project.
  - Abnormal Case:

        - Backend fetch failure: “Failed to load player list. Please try again later.”
        - No players in the project: Placeholder message displayed (e.g., “No players found.”)
    - Success Case:
        - Player list loads correctly, displaying all players in the project with accurate playtime and status.
    - Precondition
        - Developer has access to the Project Details screen.
        - Project has at least one player associated or system allows showing an empty list.
    - Postcondition
        - Player list is displayed, scoped to the selected project.
        - Developer can continue to other actions such as adding players or inspecting stats.

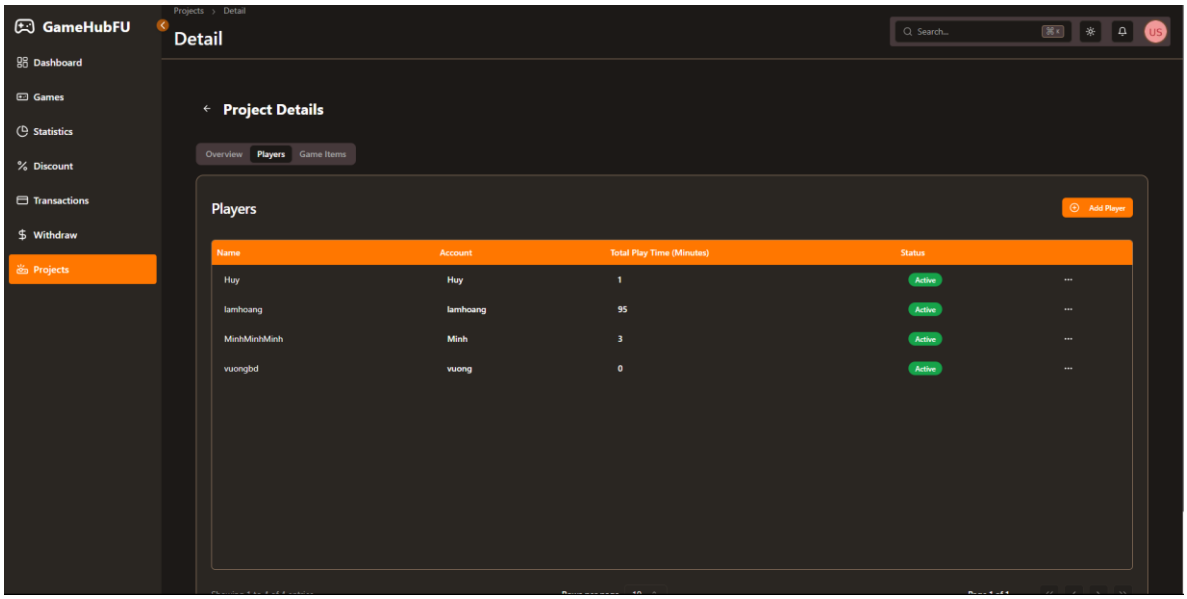

Figure 35 - View Players List Of Game Project

3.9 Game Items Of Game Project Management

3.9.1 Add New Game Item

Actor(s): Developer

- Function trigger: From the Project Details screen, the Developer selects the "Players" tab.
- Function description: This function allow “Developer” create game items for their source game project. Developer will get these item to integrate game project through integrate get all game item api.
- Function details: This function allows Developers to create a new game item for their project by filling out a form with item details such as name, type, rarity, price, image, stackability, and custom attributes. The item will be added to the selected game project upon submission.

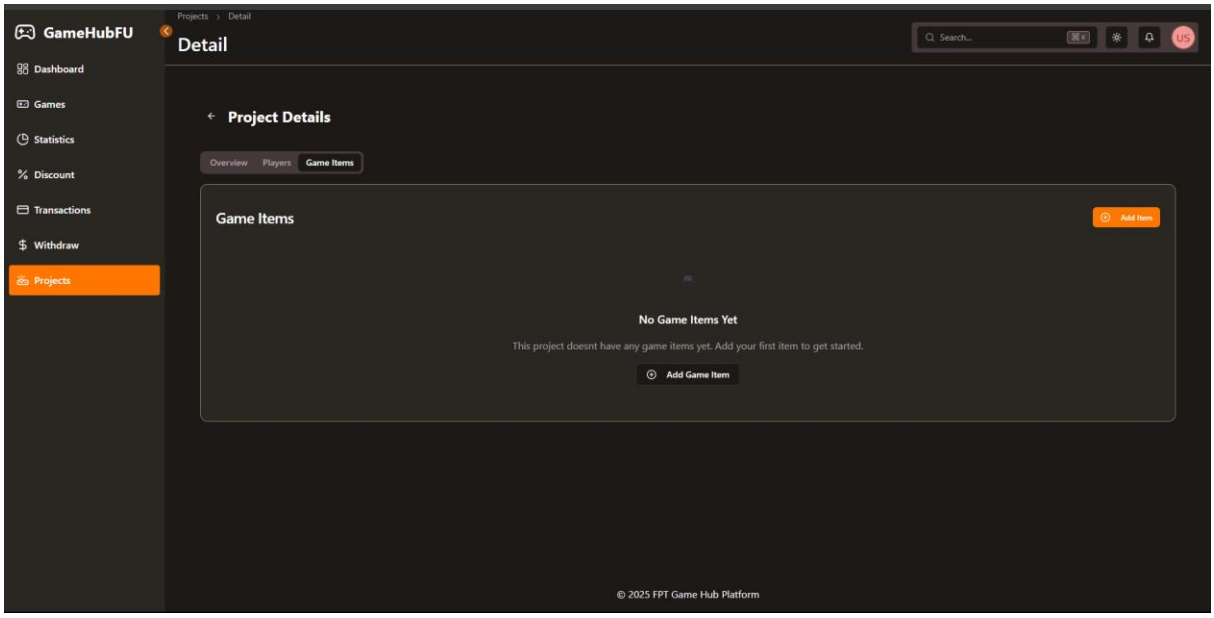

Figure 36 - View Game Items Of Game Project

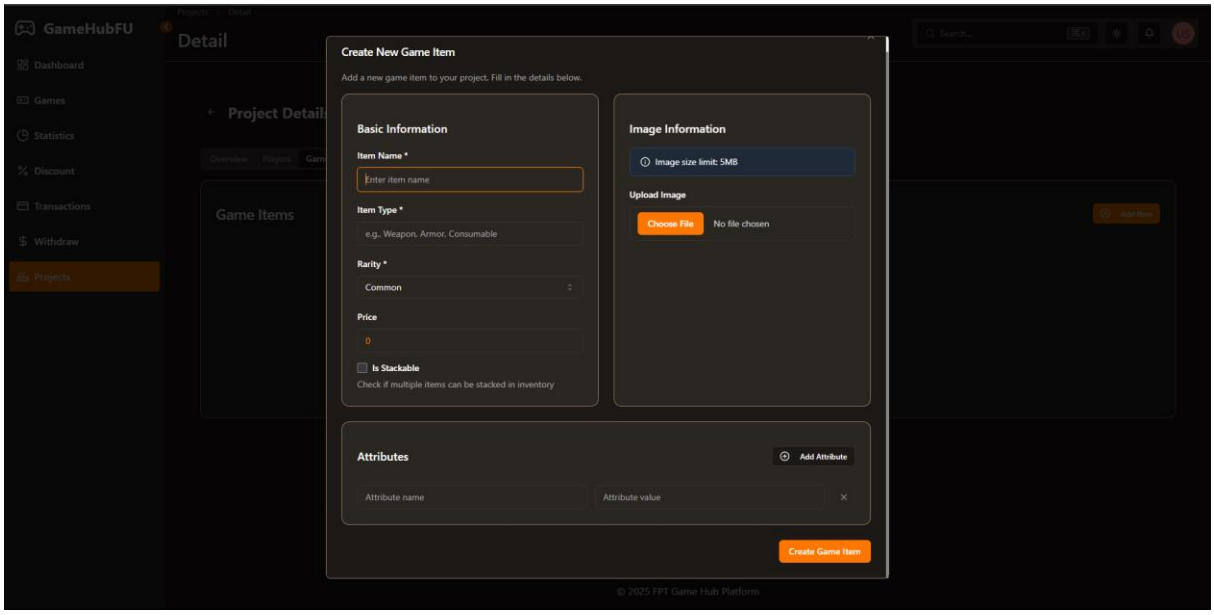

Figure 37 - Create New Game Item Of Game Project

3.9.2 Remove Game Item

Actor(s): Developer

- Function trigger: From the Project Details screen, the Developer selects the "Game Items" tab, selects one game item and select the option “Delete” from the dropdown.
- Function description: Allows a Developer to permanently remove a specific game item from a project for the purpose to manage and clean up unused or obsolete items within the game project
- Function details: This function require developer login and is the owner of game item, developer selects "Delete item" from the dropdown. Then a confirmation dialog appears to verify the deletion. Upon confirmation, the item is permanently deleted.

Figure 38 - Select Game Item To Remove From Game Project

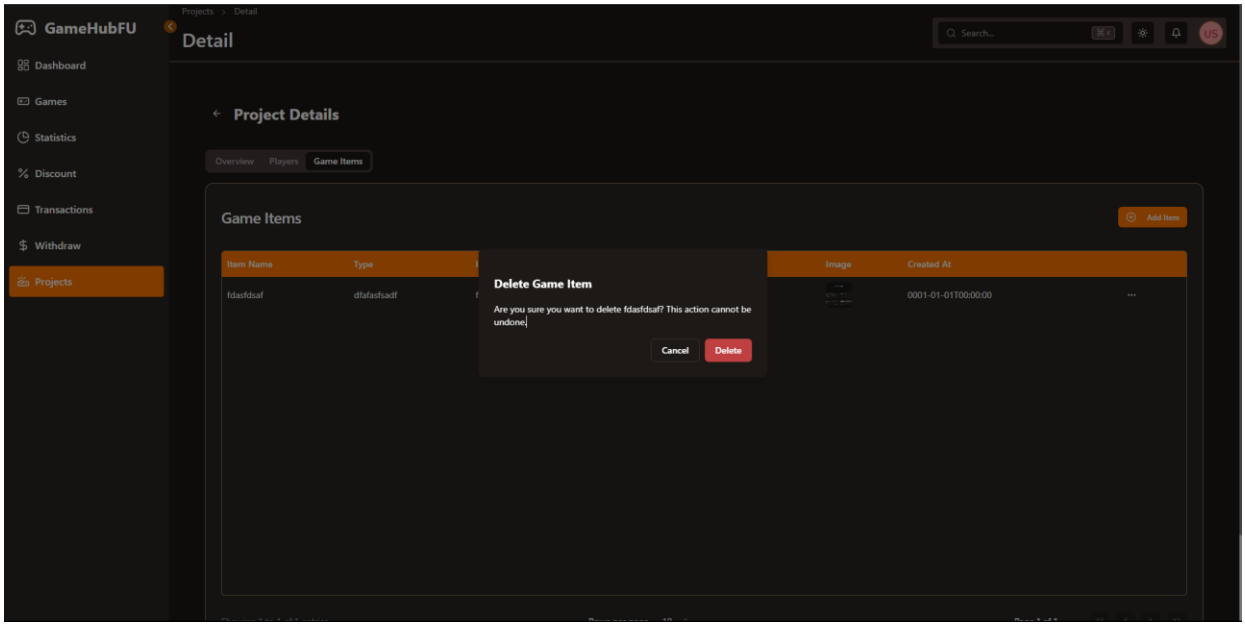

Figure 39 - Confirm Remove Game Item From Game Project

3.9.3 View All Game Items In List

Actor(s): Developer

- Function trigger: From the Project Details screen, the Developer selects the "Game Items" tab.
- Function description: This feature allows Developers to view and monitor all game items associated with a specific game project. The table provides structured data such as item name, type, stackability, price, rarity, and creation date. It helps Developers manage item data quickly and enables further actions like viewing, editing, or deleting individual items.
- Function details: When the Developer selects the "Game Items" tab, the system fetches and presents a list of game items scoped to the selected project. Each item is displayed in a row showing its name, type (e.g., Armor, Spell), whether it is stackable (true/false), price, rarity tier, thumbnail image, and creation time.

- The list is retrieved in real time from the backend and refreshed every time the Developer opens the tab to ensure accuracy.
- Each row contains an action button ("...") allowing the Developer to choose between View, Edit, or Delete (handled in separate functions). An Add Item button is displayed at the top-right corner of the screen, allowing quick creation of a new game item. The table supports sorted display by default (e.g., by creation date), and can be extended later to support pagination or lazy loading for large item sets.

- Abnormal Case:
  - Backend fetch failure: “Failed to load game items. Please try again later.”
  - No items in the project: Placeholder message is displayed: “No game items found.”
  - Missing or inconsistent data (e.g., item has undefined rarity): “Some item information is unavailable. Please refresh.”
- Success Case:
  - Game items are loaded and displayed correctly, scoped to the selected project. Developers can browse through all items, confirm their details (e.g., price, rarity, stackability), and perform further actions via the dropdown menu.
- Precondition:
  - Developer has access to the Project Details screen.
  - Project has at least one game item associated or the system allows displaying an empty list.
- Postcondition:
  - Game item list is displayed, scoped to the selected project.
  - Developers can continue with actions such as adding or modifying game items.

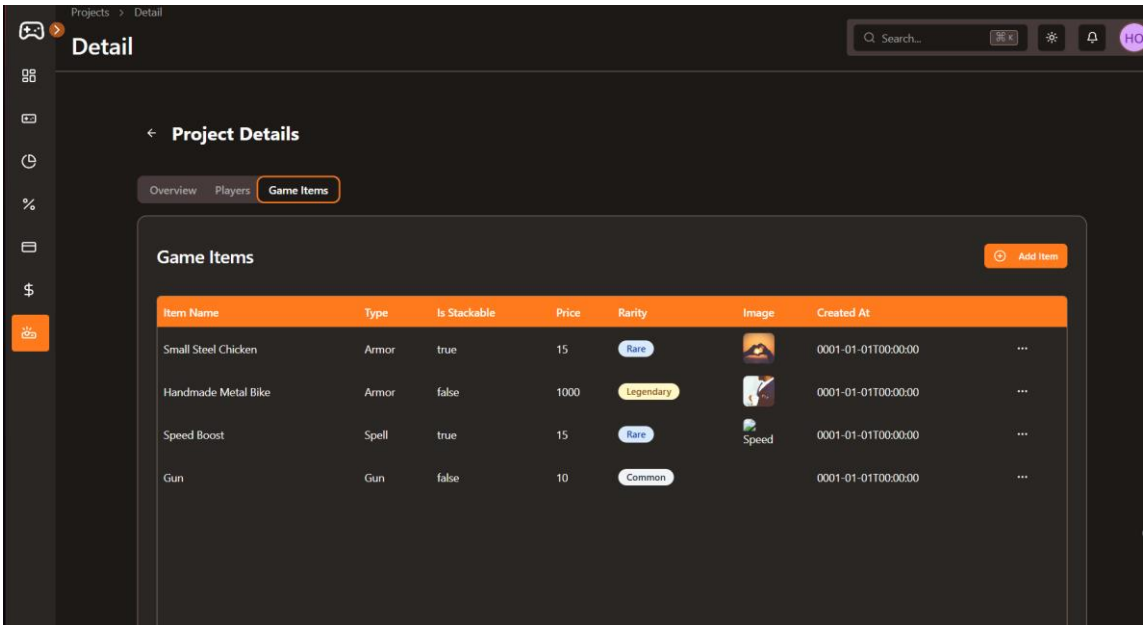

Figure 40 - View Game Items Of Game Project

3.9.4 View Game Item Information

Actor(s): Developer

- Function trigger: From the Project Details screen, the Developer selects the "Game Items" tab → Clicks on the "..." (three dots) button on a game item row → Selects View item details from the dropdown menu.
- Function description: This feature allows Developers to view complete details of a specific game item in a read-only modal window. The modal provides a structured overview of item metadata, attributes, and image-related information, enabling quick inspection without navigating away from the current screen. This helps Developers verify item data and confirm design or configuration decisions.
- Function details: When the Developer selects View item details, the system opens a modal displaying full information about the selected game item.
    - The modal includes a large preview image, item metadata (e.g., name, type, rarity, ID), and two grouped sections: Attributes and Image Information. The modal content is scrollable if necessary and presented in a clean, readable layout.
    - All data is retrieved in real time to reflect the latest item state. If the game item was modified or deleted in the background, the modal may fail to open or reflect stale data. In such cases, the system handles errors gracefully.
    - The Attributes section lists all defined properties (e.g., Damage, Speed). If the item has no attributes, a placeholder message is displayed. The Image Information section displays metadata like title, alt text, format, and media ID (if present).
    - This function is strictly read-only; no editing actions are available within the modal. Developers can close the modal using the close (X) icon at the top right.
    - Abnormal Case:
        - Item not found or deleted before modal opens: “Failed to load item details. The item may have been removed.”
        - No attributes defined: “No attributes defined for this item.”
        - Image metadata partially missing: Missing fields are shown as blank or with a default placeholder.
    - Success Case:
        - The modal opens and displays all item details accurately. Developers can verify information such as type, rarity, and image format without leaving the Project Details screen.
    - Precondition:
        - Developer is viewing the Project Details screen and has access to the Game Items tab.
        - The selected game item exists and belongs to the current project.
    - Postcondition:
        - The game item’s details are displayed in a modal.
        - Developer can review the item and then close the modal to return to the item list.

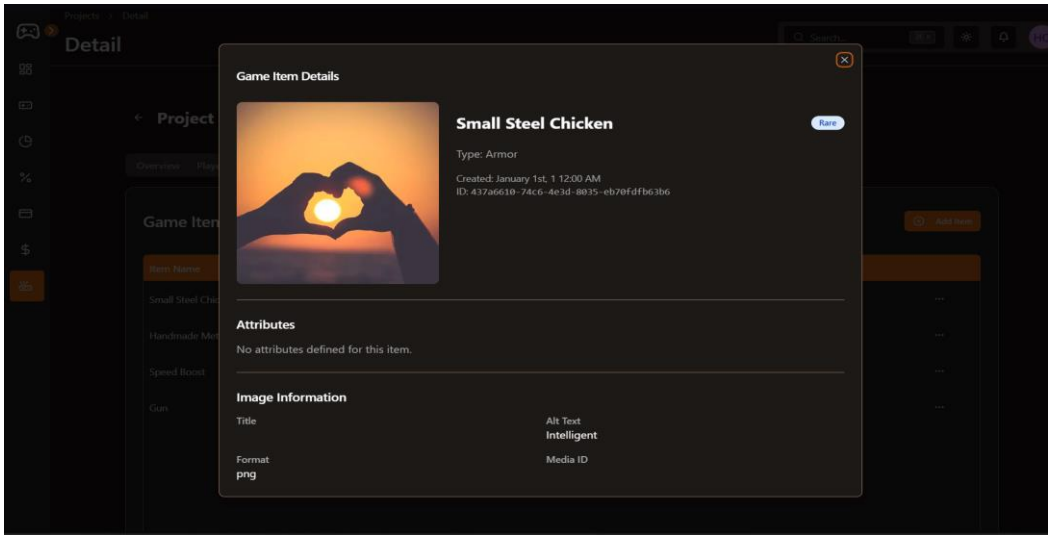

Figure 41 - View Game Item In Details

3.9.5 Integrate Api Purchase Game Items By Using Game Hub Account In Source Game Project

Actor(s): Developer

- Function trigger: Developers integrate the provided Purchase API into their Source Game Project, enabling players to buy in-game items using their Game Hub account balance.
- Function description: This feature allows Developers to connect the Game Hub purchase system into their games, so that players can buy items directly using their in-platform balance. The transaction is confirmed through a modal and processed via a backend API.
- Function details: This function handles in-game purchase transactions initiated from the Source Game. Upon confirmation, the system sends an encrypted API request to Game Hub containing purchase details such as player ID, item ID, quantity, and unit price. The backend validates the player's balance and item eligibility, calculates total cost, deducts coins, records the transaction, and delivers the item. The result is reflected both in the player's in-game inventory and Game Hub account. API responses include success (item added) or failure (e.g., insufficient balance, invalid item).
    - Abnormal Cases:
        - API failure (e.g., network issues, invalid data): "Purchase could not be completed. Please try again later."
        - Insufficient balance: "Not enough coins. Please top up."
        - Canceled transaction: No action taken.
    - Success Case:
        - Player’s coins are deducted, and purchased items are added to their in-game account. The transaction is logged in Game Hub.
    - Precondition:
        - Players must be logged in with a valid Game Hub account. The Developer must have integrated the purchase API.
    - Postcondition:
        - Coins are deducted, items are delivered, and the transaction is successfully recorded.

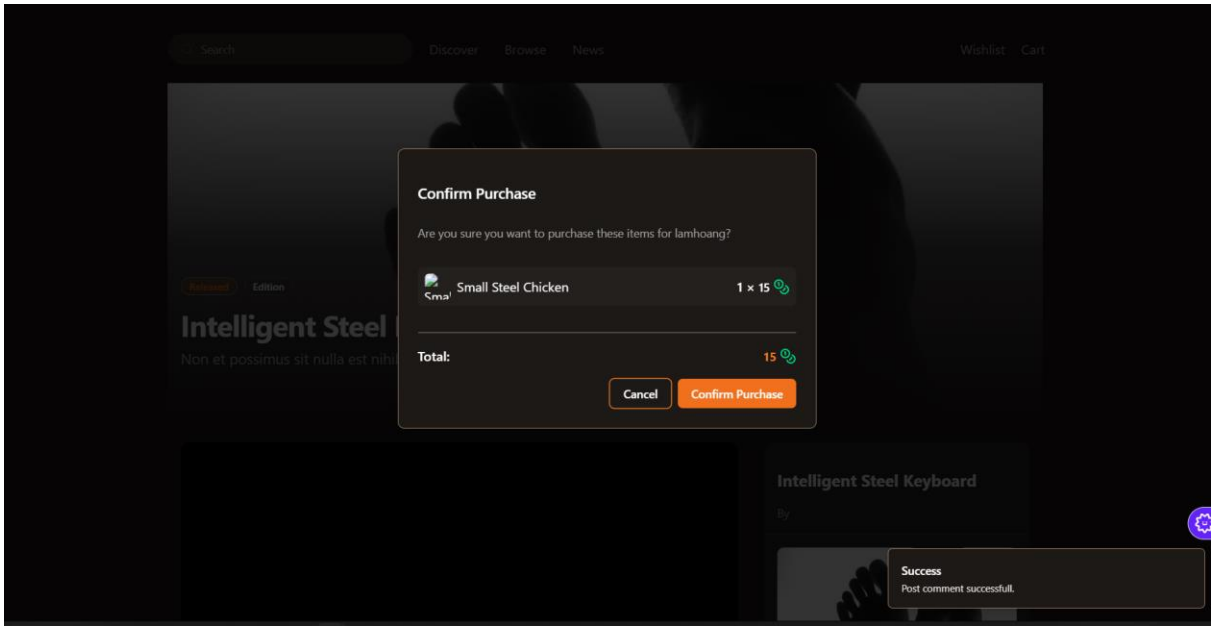

Figure 42 - Confirm Purchase Game Item

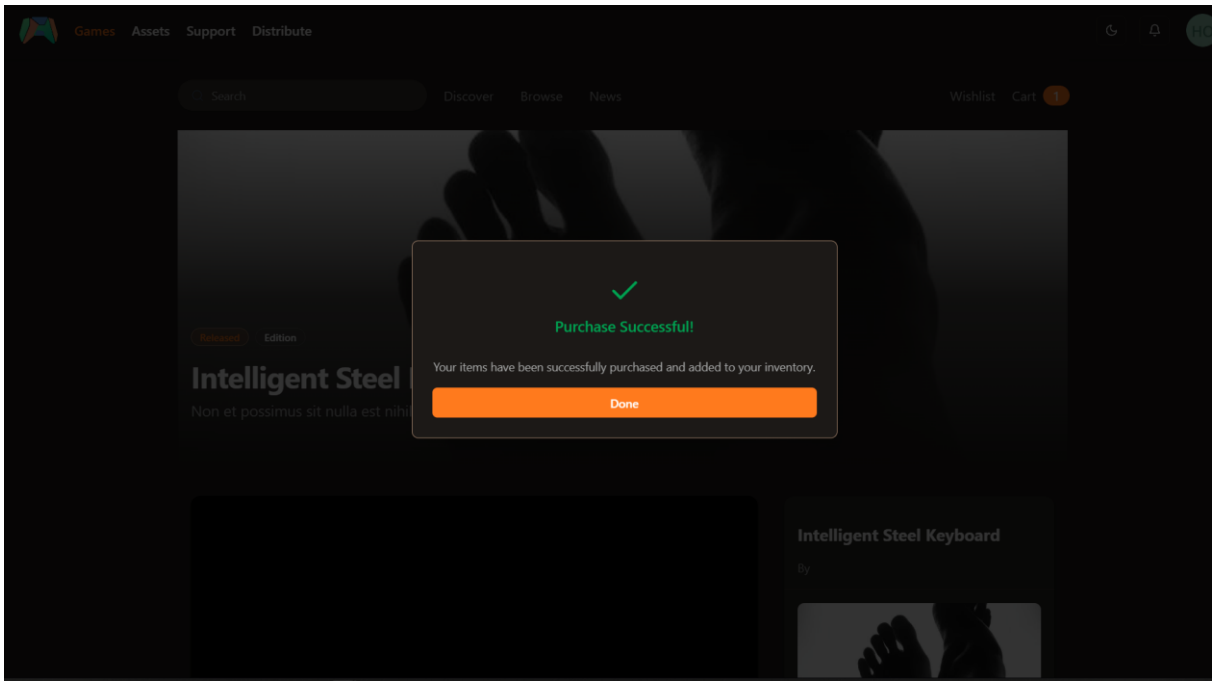

Figure 43 - PopUp Notify Purchase Game Item Successfully

3.9.6 View Purchased Game Items Of Game Player In Developer GameProject By Integrating Game Items API

Actor(s): Developer

- Function trigger: Developer implements the provided View Purchased Game Items API into their Source Game Project, enabling players to retrieve their currently owned game items via their Game Hub account.
- Function description: This feature allows the Source Game to fetch and display the list of game items owned by a player, using their Game Hub account ID. It helps synchronize the player's inventory from the Game Hub system into the game client at runtime.

- Function details: This function retrieves the player's full list of owned game items from Game Hub. The Source Game sends a GET request to the /api/PlayerGameItems endpoint, including the player's Game Hub account ID as a query parameter. The API responds with an array of item records, each containing information such as item name, type, rarity, image URL, and quantity. Developers are responsible for determining how the item data is displayed inside the game (e.g., inventory screen, item selector, or loadout system). The API is read-only and returns real-time ownership data for the given player.
    - Abnormal Cases:
        - Invalid or missing player ID: The API returns a 400 or 404 error. The Source Game should display fallback messages like "Unable to retrieve inventory. Please try again."
        - No items owned: API returns an empty array []. The Source Game should display a message such as "No items found. Start collecting now!"
        - Temporary network issues: Retry logic is recommended to ensure reliability.
    - Success Case:
        - The player's owned items are successfully retrieved and can be displayed in the Source Game. Each item includes metadata like name, type, rarity, quantity, and image URL. Developers can sort or organize these items as needed.
    - Precondition:
        - The player must be authenticated and linked with a valid Game Hub account before calling the API. The Developer must have integrated the inventory retrieval API.
    - Postcondition:
        - The Source Game receives and optionally caches the player's inventory data. Developers can use this data to power inventory screens, item selectors, or equip systems.

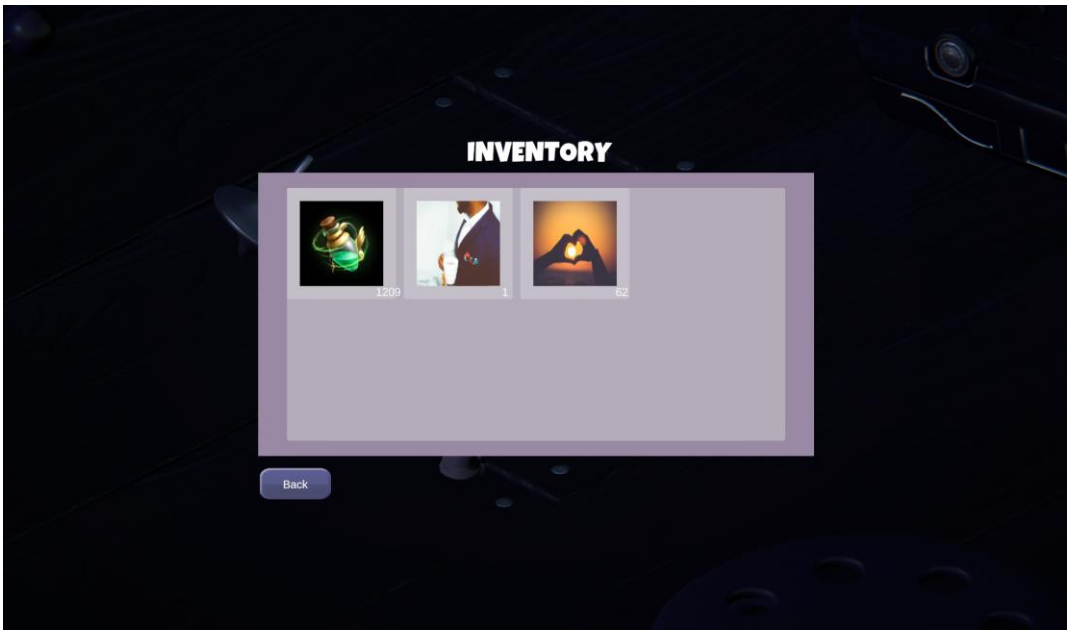

Figure 44 - Get Game Items In Developer's Game Project

3.10 Game Product Management

3.10.1 Add New Game Product

Actor(s): Developer

- Function trigger: Developer clicks the "Create Game" button located at the top right of the Games section.
- Function description: The Add New Game Product functionality allows the Developer to create a new game product on the platform. This process involves providing detailed information about the game, including the game's name, developer, publisher, support information, supported languages, categories, and player modes. The Developer can also add social media links related to the game for community engagement. Once all the necessary information is provided, the Developer can submit the game for review.
- Function details: The function return a Create New Game page where the Developer can input details about the new game by input these following information:
    - Basic Information
    - Player Modes
    - Categories
    - Supported Languages
    - Social Media Links

    Then the developer clicks the “Create” button to submit the game.

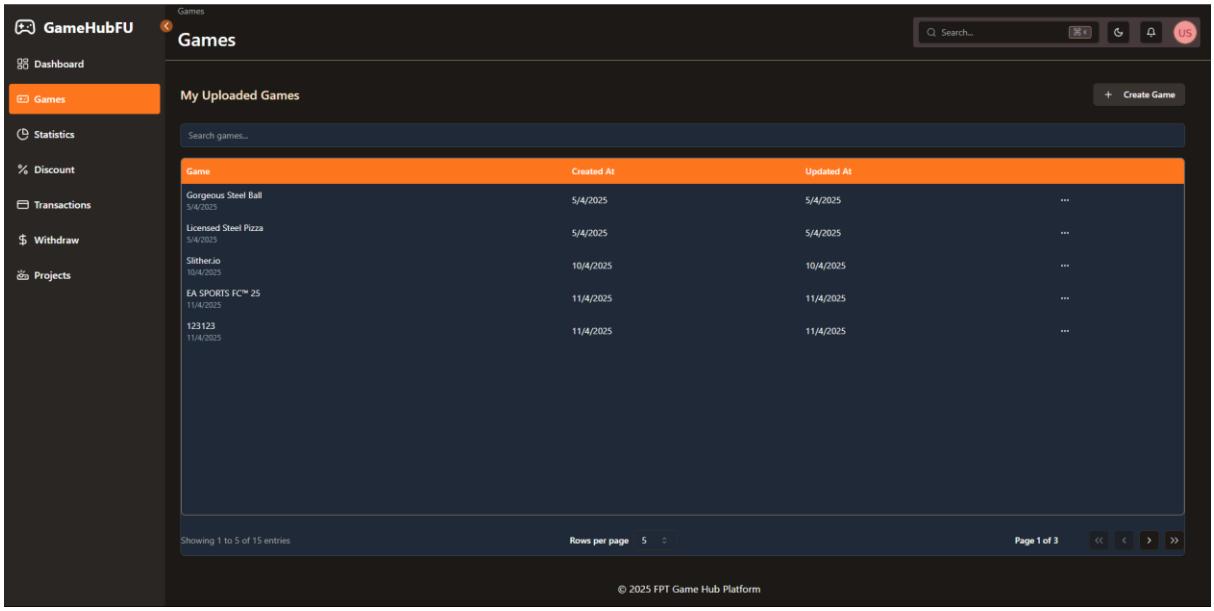

Figure 45 - Entrance Create Game Product

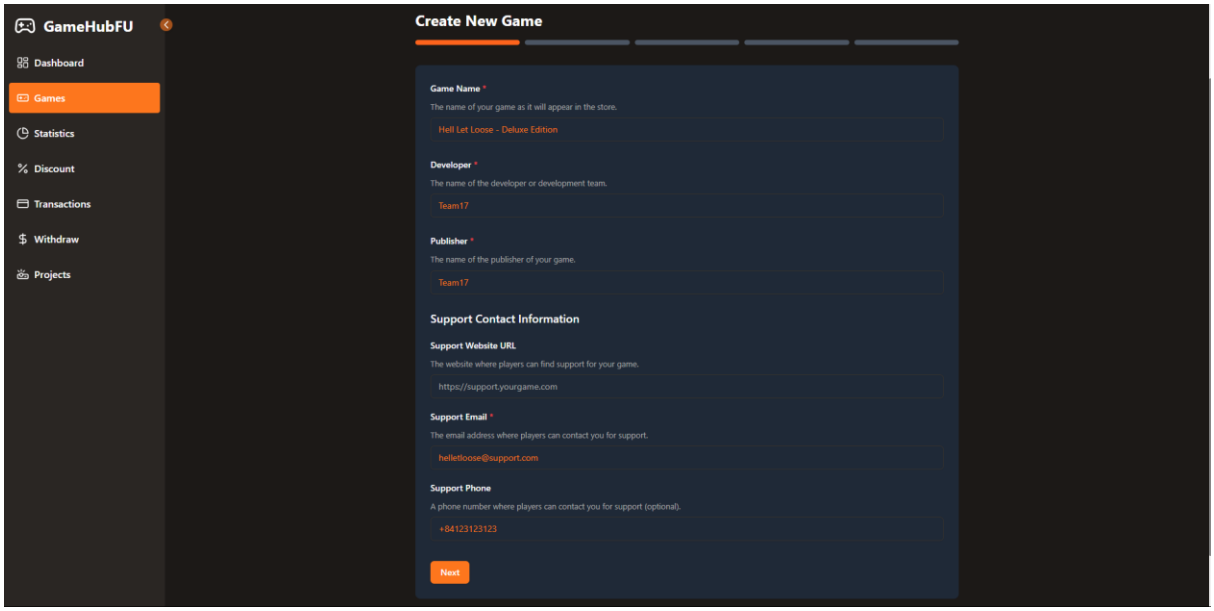

Figure 46 - Form Create Game Product

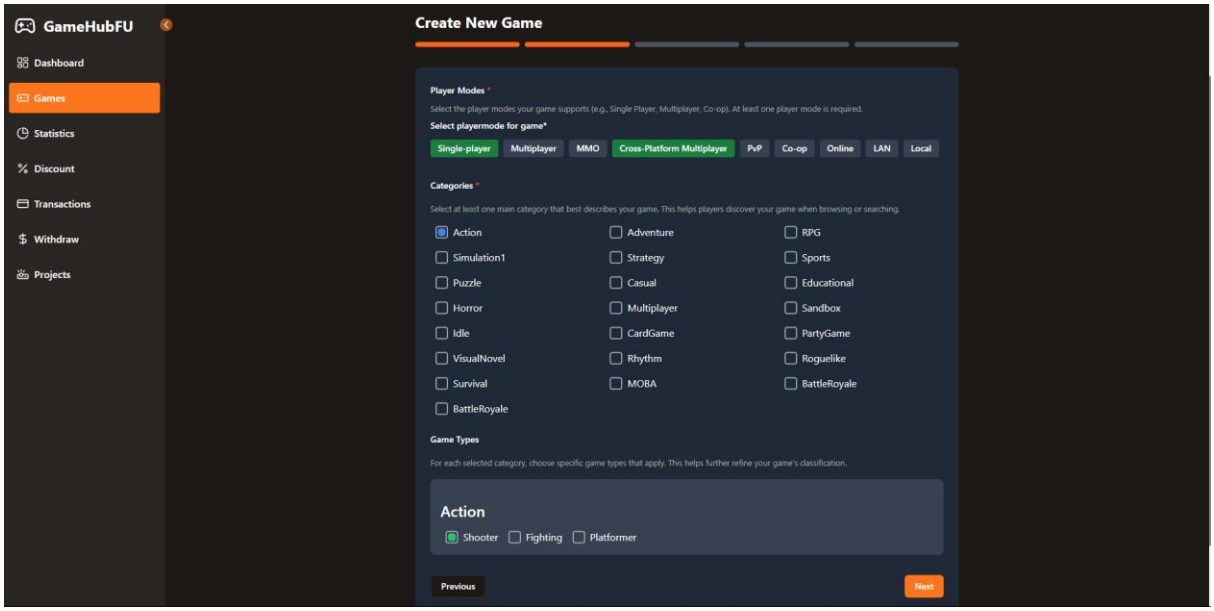

Figure 47 - Add Categories To Game Product

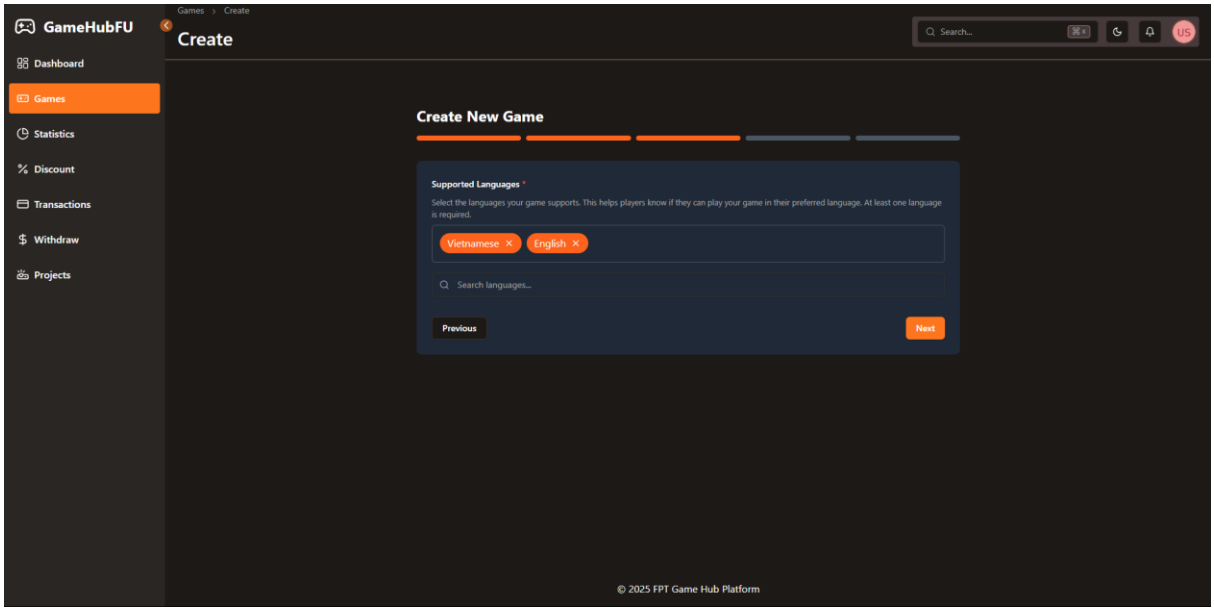

Figure 48 - Add Language Game Product Uses

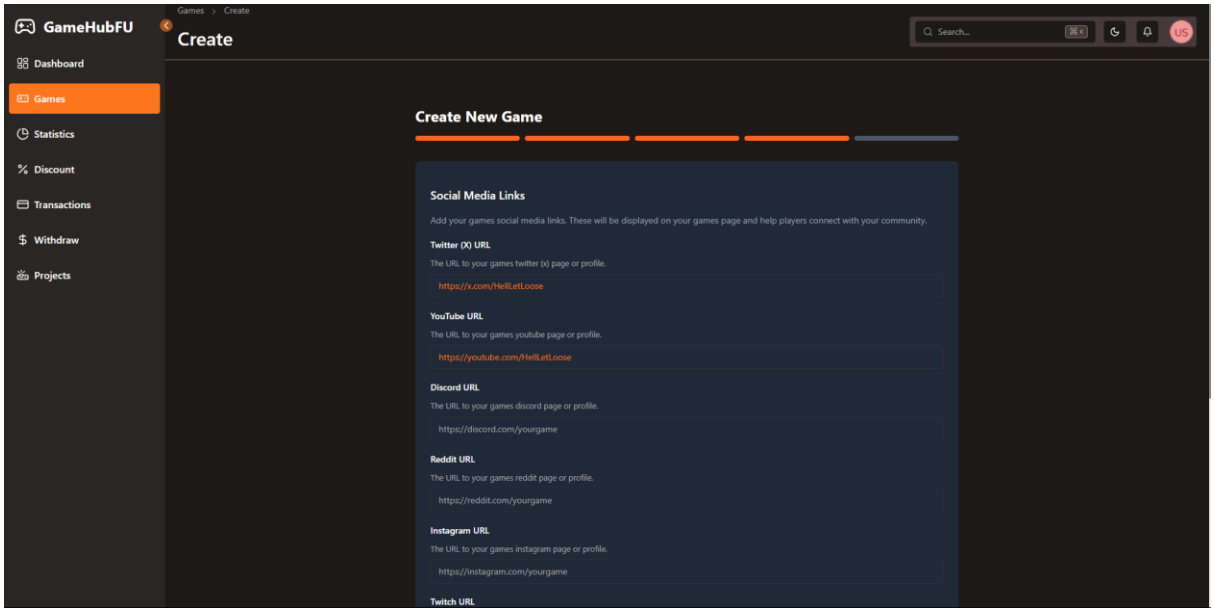

Figure 49 - Add Social Media To Game Product

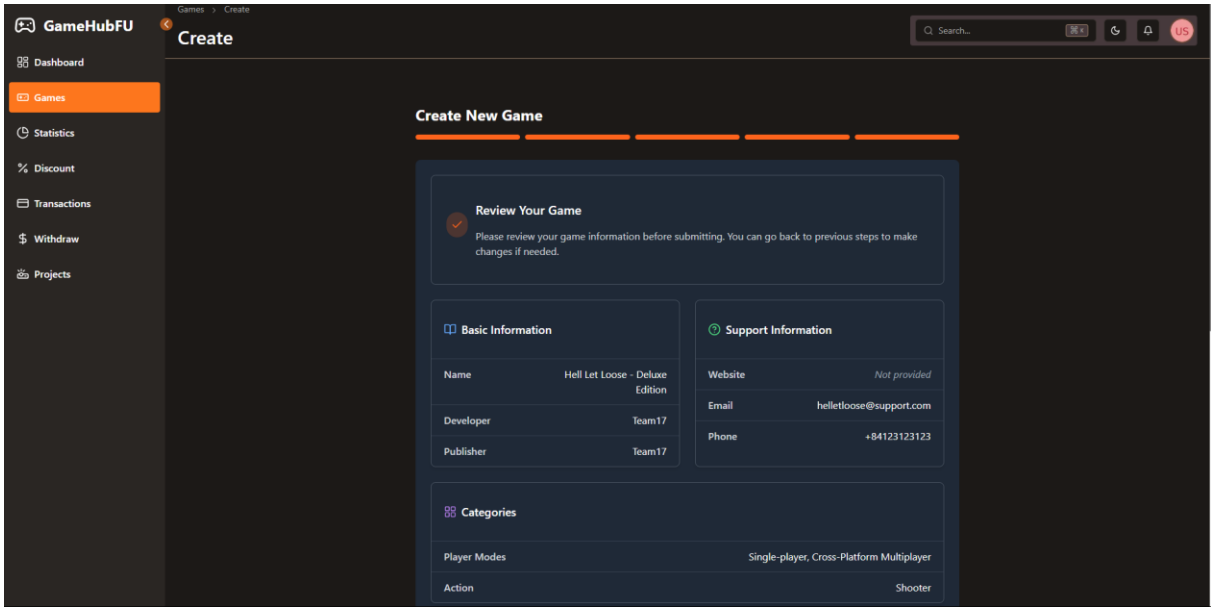

Figure 50 - Preview Game Product Before Submitting

3.10.2 Remove Game Product

Actor(s): Developer

- Function Trigger: The Developer clicks on the "Delete Game" option in the dropdown menu next to the game they wish to remove.
- Function Description: The **Remove Game Product** functionality allows the **Developer** to permanently delete a game from their uploaded game list. This action will also delete any associated game packages. Once deleted, the action cannot be undone, and the game will no longer be available on the platform.
- Function Details: This triggers a confirmation dialog to verify if the **Developer** wants to permanently delete the game.
    1. Open the Game List: The Developer navigates to the Games section where they can see a list of their uploaded games, including the game title, creation date, and last update date.
    2. Trigger the Deletion: Dropdown Option: From the dropdown menu, the Developer selects the "Delete Game" option.
    3. Confirm the Deletion:
        - Dialog Box: A confirmation dialog appears asking the Developer to confirm the deletion.
        - Message: The confirmation message will state: "Are you sure? This will permanently delete [Game Name] and all associated packages. This action cannot be undone."
        - Button/Navigation:
            - Cancel Button: If the Developer clicks Cancel, the deletion is aborted, and no action is taken.
            - Delete Button: If the Developer clicks Delete, the game is permanently removed from the platform.
    4. Completion:

- After clicking Delete, the game is permanently deleted, and the Developer is returned to the My Uploaded Games page, where the game is no longer listed.

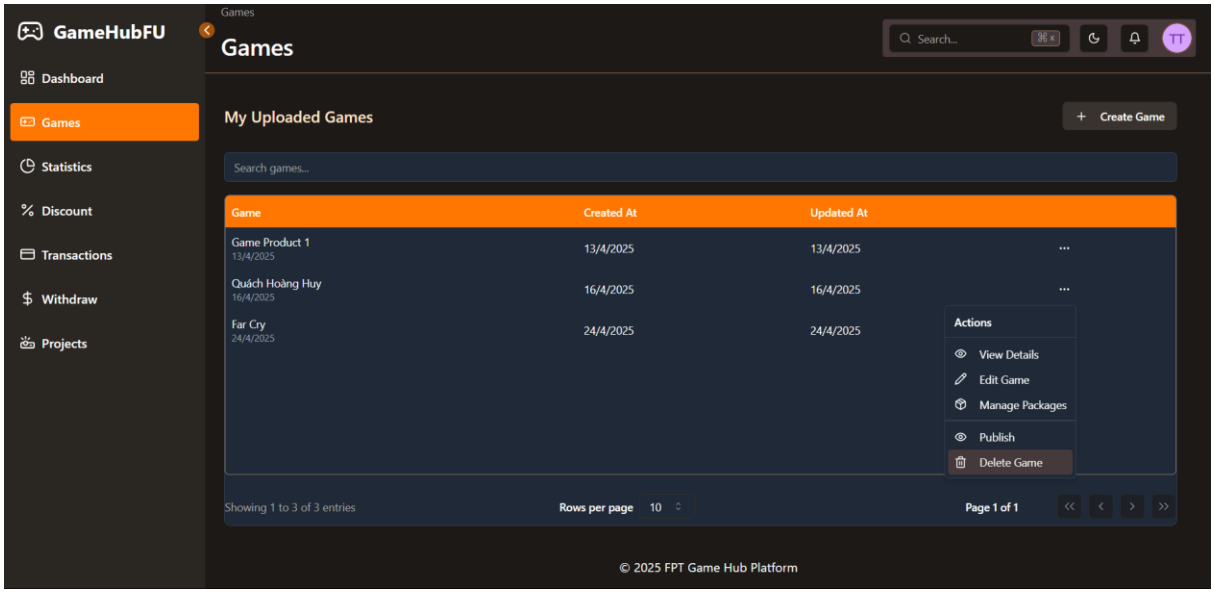

Figure 51 - Select Game Product To Remove

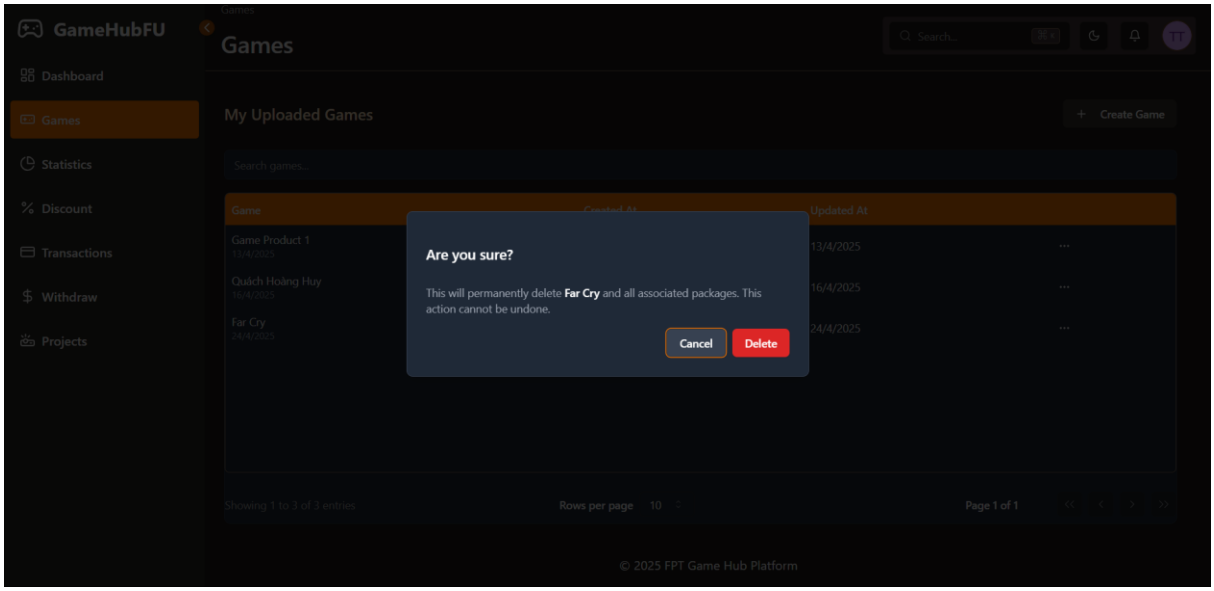

Figure 52 - Confirm Delete Game Product

3.10.3 View All Game Product In List

Actor(s): Developer

- Function Trigger: The Developer opens the “Games” tab.
- Function Description: The View All Game Product in List function allows the Developer to see a list of all the games they have uploaded to the platform. The list includes basic information such as the game name, creation date, last update date, and actions that can be performed on each game. This provides the Developer with an overview of their uploaded games and their status.
- Function Details: The system displays a list of all games uploaded by the Developer in the "My Uploaded Games" section.

1. View Game Details:
    - Displayed Information:
        - Game Name: The title of each game.
        - Created At: The date the game was first uploaded.
        - Updated At: The date the game details were last updated.
2. Available Actions:
    - Actions Column: For each game listed, the Developer can take the following actions:
        - View Details: View detailed information about the game.
        - Edit Game: Edit the details of the game.
        - Manage Packages: Manage the associated packages for the game.
        - Delete Game: Permanently delete the game from the platform (with confirmation).

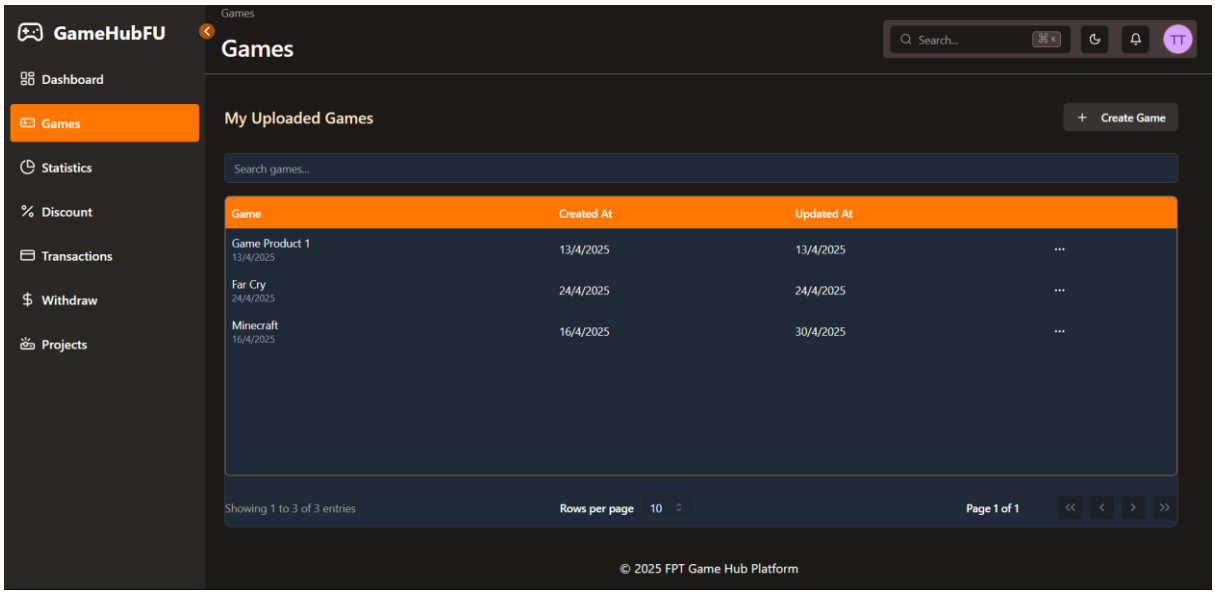

Figure 53 - View All Game Products List

3.10.4 View Game Product In Details

Actor(s): Developer

- Function Trigger: The Developer clicks on the game name or the "View Details" option in the "Actions" column
- Function Description: The View Game Product in Details function allows the Developer to access detailed information about a specific game that they have uploaded. This includes general game information, player modes, associated social media links, game packages, statistics (views, downloads), and support contact details. The Developer can review and manage these details directly.
- Function Details: The system opens the detailed view for that game, showing various sections of game information.
    1. Game Information Displayed these following information:
        - Basic Information: Game Name, Developer, Publisher, Release Date
        - Player Modes: Information on the types of gameplay supported (e.g., single-player, multiplayer, MMO, cross-platform multiplayer).

- Social Media Links: Links to the game's social media profiles (Twitter, YouTube, Discord, Reddit, etc.).
- Game Categories: The categories that describe the game (e.g., Action, Adventure, RPG, Sports, etc.).
- Game Packages: The Developer can manage the packages available for the game or create a new one if none exist.

2. Available Actions:
    - Edit Game: The Developer can edit the details of the game.
    - Create Package: The Developer can create new game packages.
    - Send Review Request: The Developer can send a request for the game to be reviewed by a Moderator for publishing.
    - Delete Game: The Developer can delete the game package (this will require confirmation).

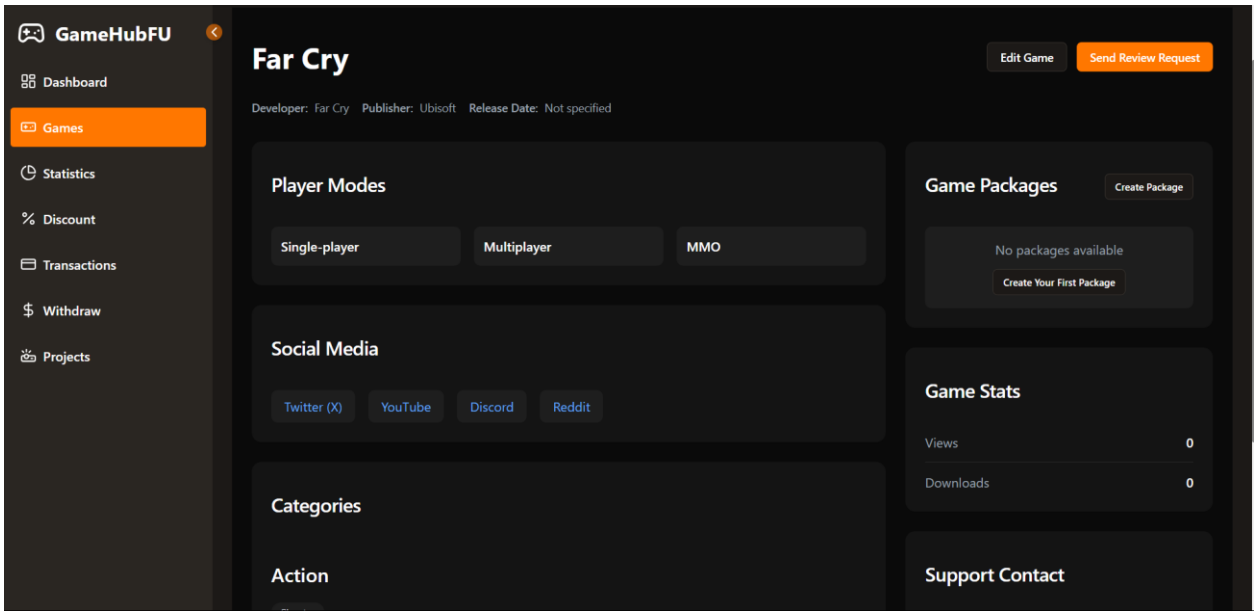

Figure 54 - View Game Product In Details

3.10.5 Upload Executable Game Builds

Actor(s): Developer

- Function Trigger: The Developer clicks on the "Create Game Package" button.
- Function Description: The Create Game Package function allows the Developer to create a new package for a game they are uploading, specifying all necessary details such as the game's basic information, descriptions, media files, executable builds, and platform requirements. The Developer will upload the game build and specify the supported platforms, completing all the necessary details to create the game package.
- Function Details: The function requires upload at least one game package source build while creating game package. The developer must pass through these steps:
    1. Fill Out Basic Information
    2. Provide Descriptions for the Game
    3. Upload Media Files
    4. Add Media Items
    5. Upload Source Files

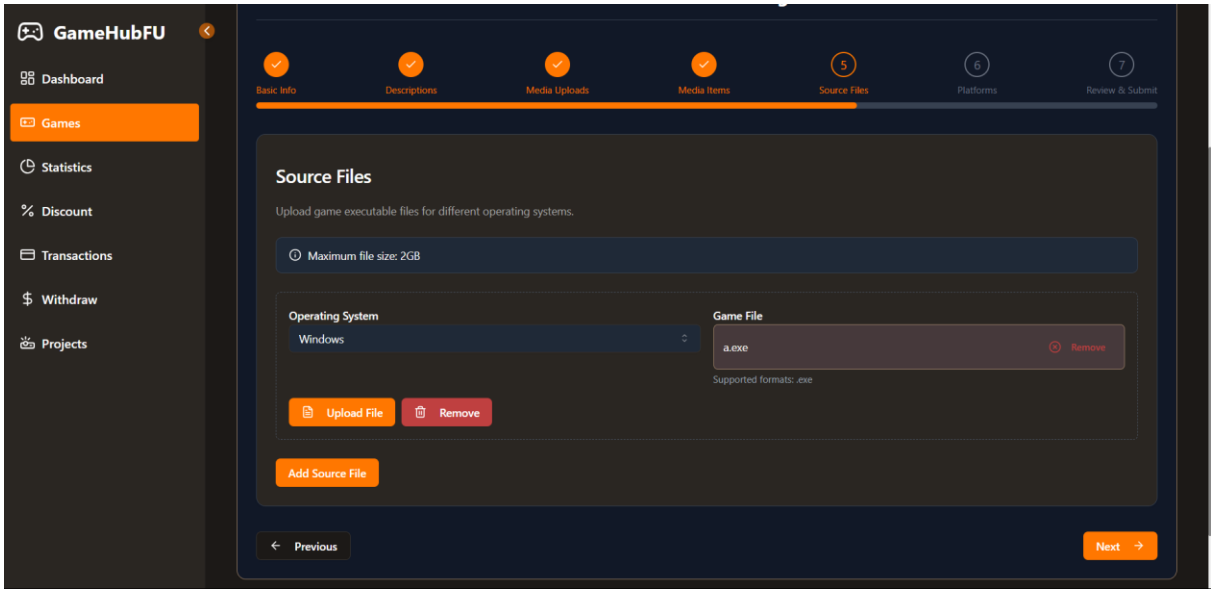

Figure 55 - Add Source Game To Game Package Of Game Product

3.10.6 Release Game Product By Now

Actor(s): Developer

- Function trigger: The Developer clicks on the "Submit for Review" button under the Review section.
- **Function Description:** The "Release Game Product By Now" function enables the Developer to submit a game package for approval by a Moderator. This is done through a review request. The game will remain in a "PendingApproval" state until the Moderator reviews and approves it for release.
- Function Details:
    - Navigate to the Review Section:
        - Button/Navigation: The Developer navigates to the "Review" tab within the game package's detail page.
        - Function: The system displays the option to submit the game package for review.
    - Submit the Game for Review:
        - Button/Navigation: The Developer clicks the "Submit for Review" button.
        - Function: The system submits the game package to the Moderator for review.
    - Review Process:
        - Button/Navigation: The game package is now marked as "PendingApproval".
        - Function: The system waits for the Moderator to approve or deny the game for release.
    - Post-Action Behavior:
        - After submitting for review, the game package is in "PendingApproval" status until the Moderator completes the review.
        - The Developer can continue to view and manage the game package but cannot release it until the Moderator approves it.

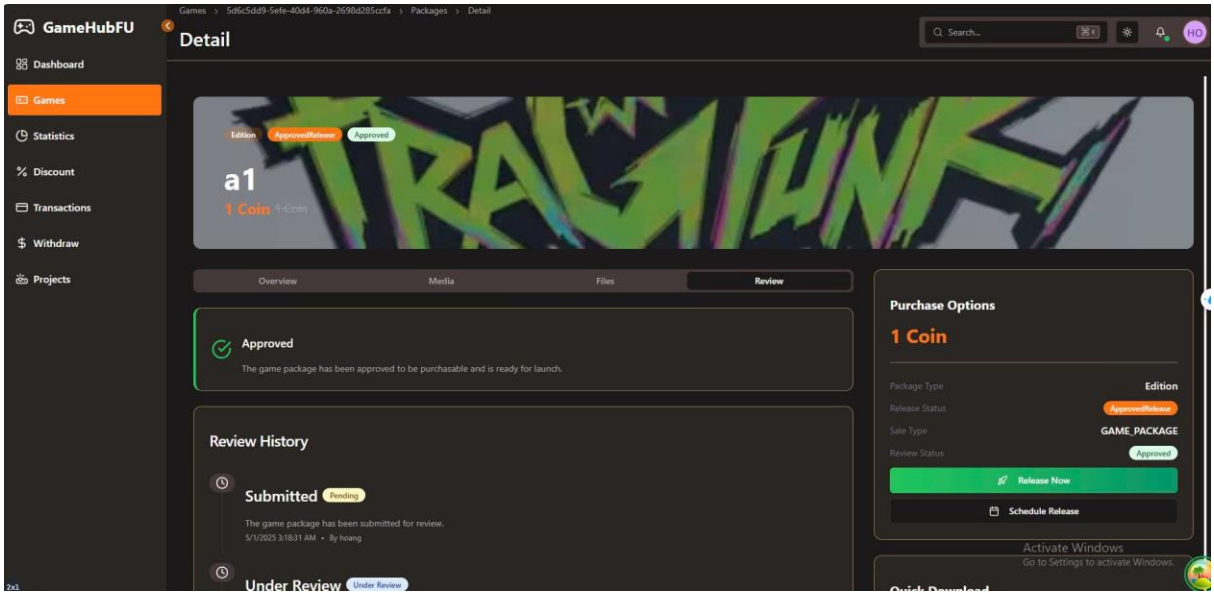

Figure 56 - Release Game Package Of Game Product

3.11 Game Asset Management

3.11.1 Add New Game Asset

Actor(s): Designer

- Function trigger: Designer adds necessary information to create a new game asset
- Function description: This feature allows Designers to create and register a new game asset in the Game Hub system. This functionality supports the game content pipeline by enabling Designers to define asset metadata, upload associated images, and categorize the item appropriately.
- Function details: This function require designer submit the form of game asset. The submission form includes fields such as asset name, product category, designer/publisher identities, social media links, tags, external references, keywords, and legal/support content. All required fields must be completed before the asset can be submitted for review or publishing.
    - Abnormal Cases:
        - Missing required fields: API returns a 400 error. Designers should be prompted to complete all mandatory inputs.
        - Server or connection failure: Designers should be given the option to save work-in-progress as a draft or retry submission.
    - Success Case:
        - The asset is successfully created and stored in the Game Hub catalog. A confirmation is shown to the Designer.
    - Precondition:
        - Designer is authenticated and authorized to manage asset content. The backend system must be ready to accept new asset records.
    - Postcondition:
        - The new game asset is saved and visible to relevant internal systems. It can now be linked to player inventory, in-game content, or reward systems as needed.

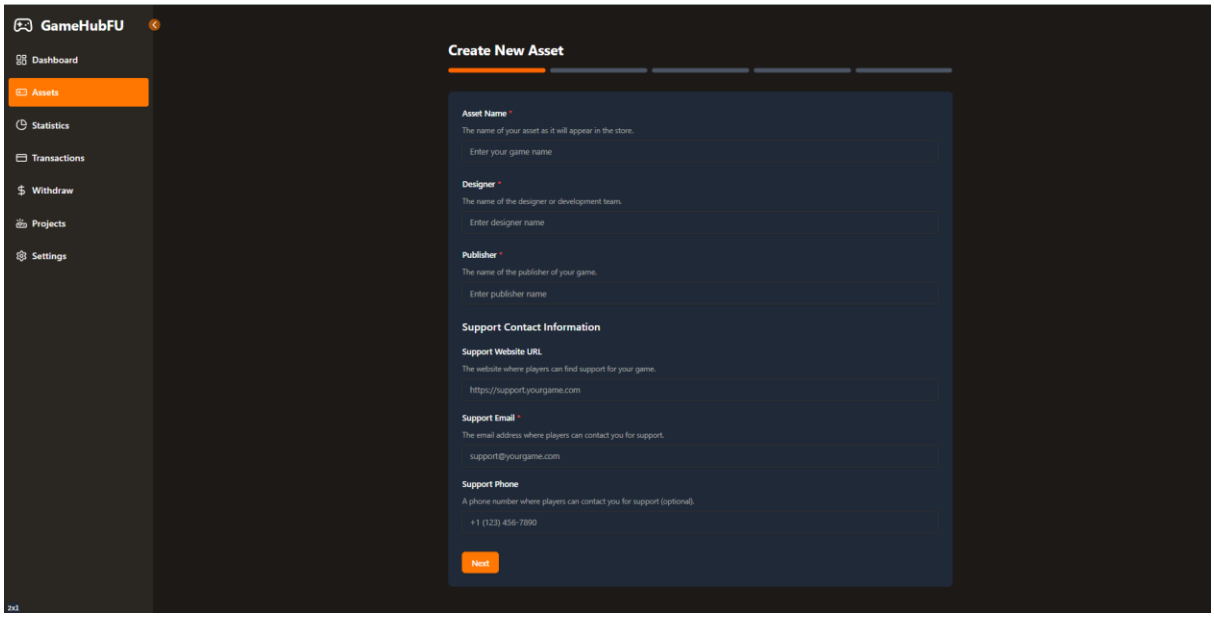

Figure 57 - Create New Asset Product

3.11.2 Remove Game Asset

Actor(s): Designer

- Function trigger: Designer selects a game asset to delete from the Game Hub system
- Function description: This feature enables Designers to remove an existing game asset from the Game Hub system by specifying its unique PackageId. It helps maintain a clean and up-to-date asset catalog by allowing obsolete, duplicated, or incorrect assets to be deleted. Once removed, the asset will no longer be available for player inventory, in-game use, or marketplace listing.
- Function details: This function requires developer has authorization to delete and is the owner of game package. Then upon deletion request, the system checks whether the asset:
    - Exists in the system
    - Is not linked to any active or dependent content
    - Has a status of "Unreleased"
    - And the requester has sufficient permissions
- If all criteria are satisfied, the asset is deleted from the platform. Otherwise, the deletion is blocked with an error message indicating the reason.
    - Abnormal Cases:
        - Missing or invalid PackageId: API returns a 400 error. The UI should prompt the Designer to enter a valid identifier.
        - Asset not found: API returns a 404 error. Designer should be shown a message like “Asset not found or already deleted.”
        - Unauthorized attempt: API returns a 403 or 401 error if the Designer is not authenticated or lacks permissions.
        - Asset is in active use: API may return a 409 Conflict if the asset is currently linked to live systems (e.g., store listings or inventory). Designers should be instructed to unlink the asset first.
    - Success Case:
        - The asset is successfully deleted from the Game Hub system. A confirmation message is shown to the Designer, and the asset is no longer accessible through the platform.

- Precondition:
  - Designer is authenticated and has appropriate permissions to manage and remove game assets. The asset identified by PackageId must exist and not be in active use.
- Postcondition:
  - The asset is permanently removed from the Game Hub catalog and is no longer available for player assignment, store listings, or in-game features.

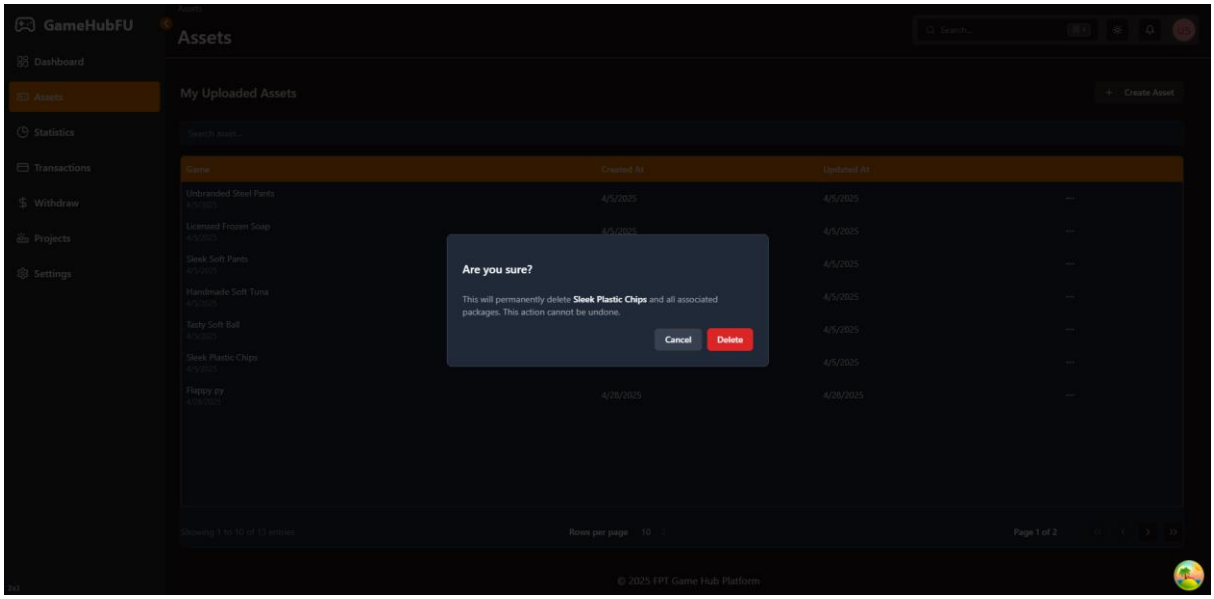

Figure 58 - Remove Asset Product Confirmation

3.11.3 View All Game Asset In List

Actor(s): Designer

- Function trigger: Designer opens the asset management dashboard to view a list of assets.
- Function description: This feature allows Designers to retrieve and view a complete list of all game assets that have been created and registered in the Game Hub system. It provides an overview of available assets for browsing, editing, or administrative actions.
- Function details: The function returns a paginated or complete list of asset records, each containing attributes such as name, category, designer, publisher, tags, and status. Designers can filter, search, or sort the list based on various fields (e.g., keyword, creation date, or type).
  - Abnormal Cases:
    - Network failure or server error: API may return 500. UI should show a message like “Unable to fetch assets. Please try again later.”
    - Unauthorized access: API returns a 401 or 403 if the Designer is not properly authenticated.
    - Empty asset catalog: API returns an empty array []. Show message: “No assets found. Start by creating your first game asset.”
  - Success Case:
    - Designer successfully views a structured list of all existing game assets, each showing essential metadata. The UI can offer filtering, editing, or deletion controls per entry.
  - Precondition:
    - Designer is authenticated and authorized to access the asset catalog.

- Postcondition:
  - The list of game assets is displayed in the UI and can be used for further actions such as editing, deleting, or reviewing asset details.

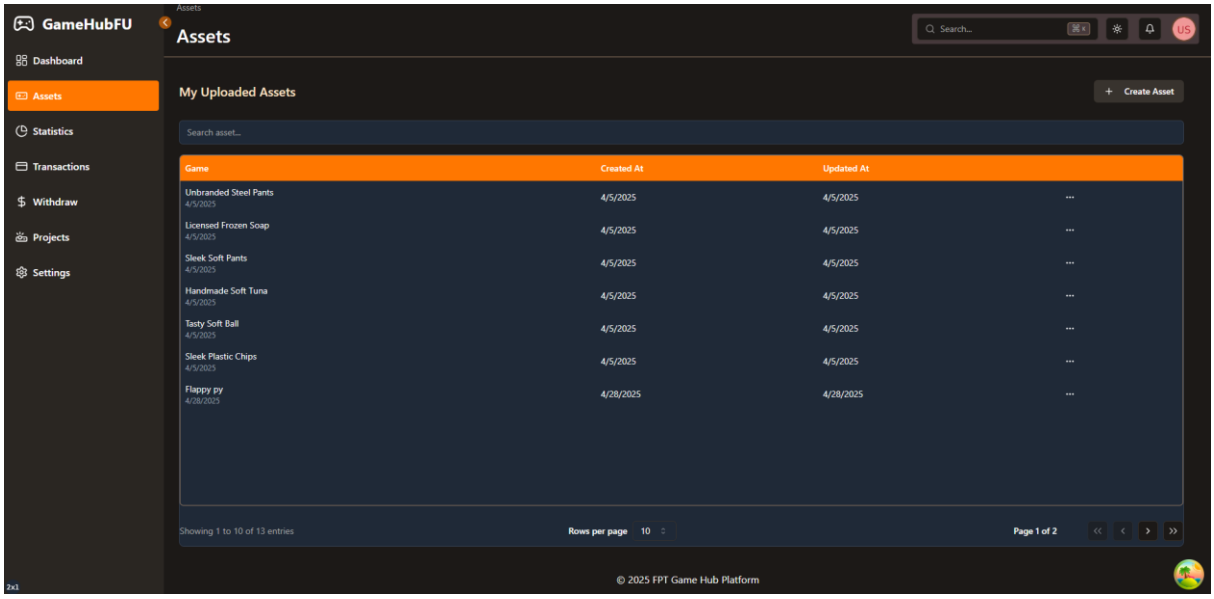

Figure 59 - View The Current Asset Products

3.11.4 View Game Asset In Details

Actor(s): Designer

- Function trigger: Designer selects a specific game asset from the asset list to view its detailed information.
- Function description: This feature allows Designers to view the full metadata and configuration of a single game asset by its unique identifier (UUID). It helps Designers verify, review, or prepare the asset for publishing, updates, or integration.
- Function details: This function returns the package with tored fields for the asset, such as category, designer, publisher, asset product type info, tags, legal lines, social media links, external links, support contacts, and timestamps.
  - Abnormal Cases:
    - Invalid or missing UUID: API returns 400 Bad Request. UI should prompt: “Invalid asset identifier.”
    - Asset not found: API returns 404 Not Found. UI should show: “Asset not found or has been deleted.”
    - Unauthorized access: API returns 401 or 403. The Designer should be redirected to login or shown an access error.
  - Success Case:
    - Designer receives and views a complete, structured view of the selected asset’s details. Data can be rendered in a read-only form or enriched with edit options depending on permissions.
  - Precondition:
    - Designer is authenticated and has permission to access detailed asset records. The asset with the given UUID must exist in the Game Hub system.
  - Postcondition:
    - The asset detail page is loaded and populated with all associated metadata. Designers can use this view to confirm content or initiate further actions like edit or delete.

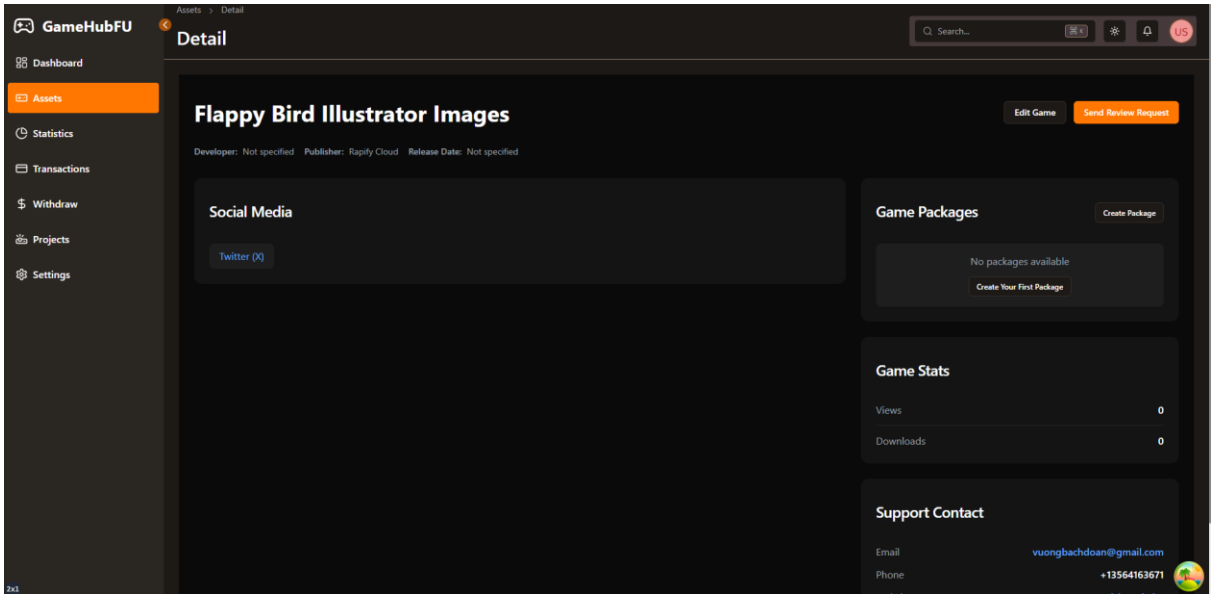

Figure 60 - View Asset Product In Details

3.11.5 Upload Asset Files

Actor(s): Designer

- Function trigger: Designer clicks the Create Package button from either three vertical dot > Manage Package or three vertical dot > View Details > Create Package
- Function description: This feature allows Designers to upload files and media assets associated with a game asset by creating an asset package. It supports uploading preview media (images and videos), descriptions, pricing, and source files to be bundled as a package linked to a game product. Function details:
    - When a Designer initiates the upload process through the UI, the system collects and sends asset files with necessary information to the backend.
    - Abnormal Cases:
        - Missing or invalid gameProductId: API returns 400 Bad Request. UI should prompt: “Invalid game asset. Please go back and try again.”
        - Upload errors (e.g., large file, unsupported format): Show alert such as “File upload failed. Please check the format and size.”
        - Unauthorized access: API returns 401/403. Prompt Designer to re-authenticate or display access restriction.
    - Success Case:
        - The asset files are successfully uploaded and associated with the specified game product. The API returns a success message or asset package ID. Designers see a confirmation and the uploaded package is listed under the associated asset.
    - Precondition:
        - Designer is authenticated.
        - Game asset (game product) already exists with a valid UUID.
        - Files to upload are prepared and meet format/size constraints.
    - Postcondition:
        - New asset package is created and associated with the game product.
        - Uploaded media and source files are stored and available for display, preview, or download.

- The package is visible in the asset management or detail view UI.

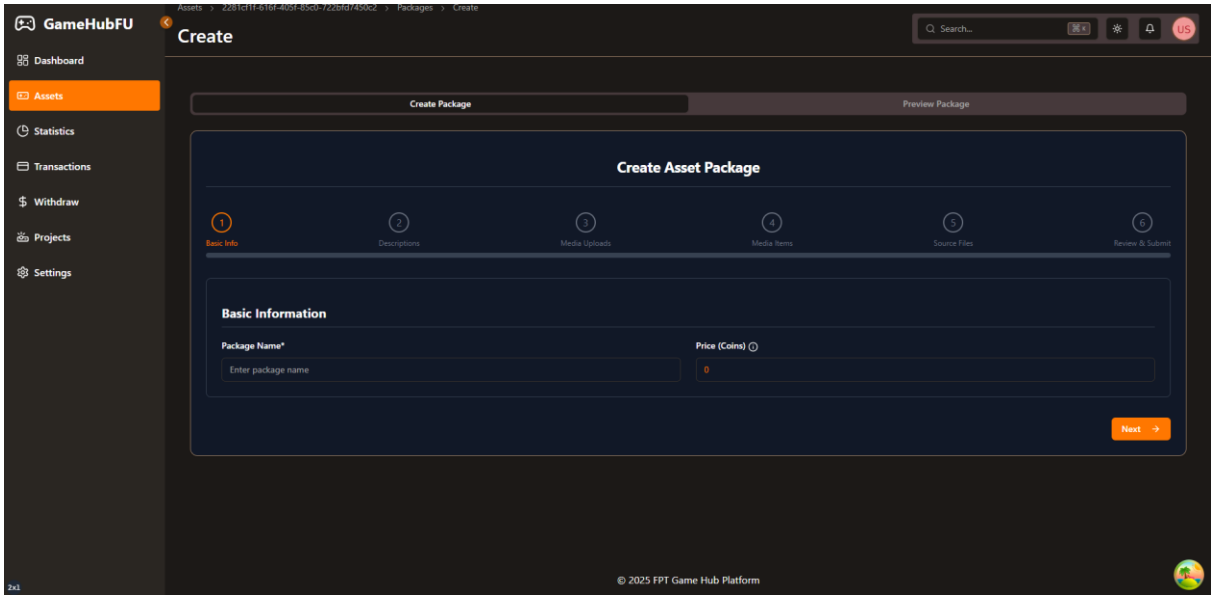

Figure 61 - Create New Asset Package Of Asset Product

3.11.6 Release Game Asset By Now

Actor(s): Designer

- Function Trigger: The Designer clicks on a specific asset listed as “Ready for Release”
- Function Description: The “Release Game Asset By Now” function allows Designers to immediately publish a prepared and validated game asset to the platform. This function is used when the asset is finalized, and the Designer wants to make it available to users without delay.
- Function Details: This function release the game asset and public on platform:
    - Release game following steps:
        1. Access the Game Asset:
            - Navigation: The Designer opens the list of managed game assets.
            - Action: Clicks on a game asset with a status of “Ready for Release”.
        2. Review Asset Details:
            - Function: The system displays detailed information including asset title, category, file size, version, and preview.
        3. Release Immediately:
            - Button/Navigation: The Designer clicks the “Release Now” button.
            - Function: The system performs final checks and changes the asset status to “Published”, making it publicly available in the store or library.
    - Release game following steps:
        - The asset status updates to Published.
        - Users on the platform can now access or purchase the asset.
    - Abnormal Cases:
        - If the asset is missing required metadata or files: *“Asset cannot be released. Please complete all required information.”*
        - System error: *“Failed to release asset. Please try again later.”*

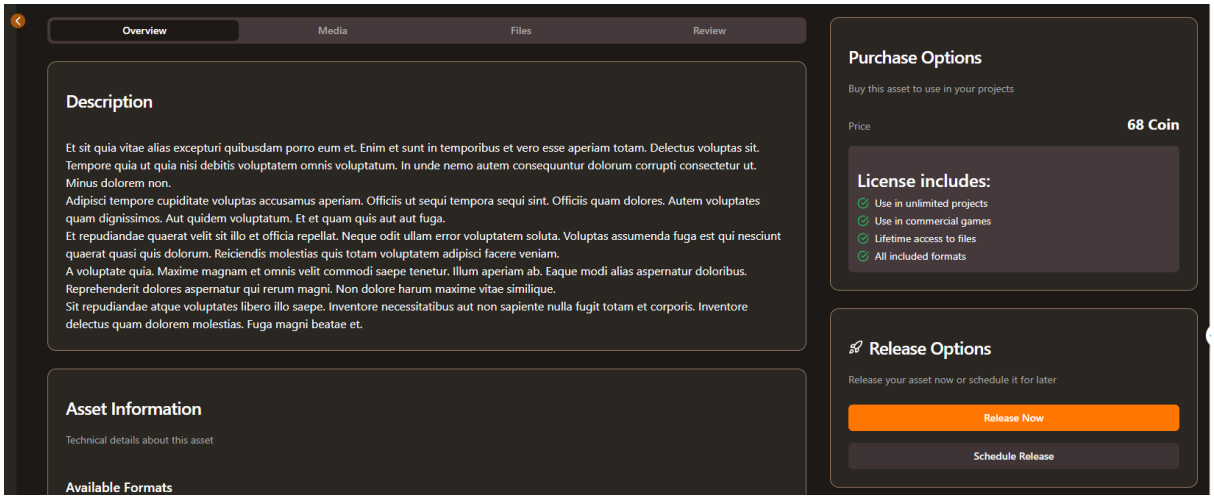

Figure 62 - Relase Asset Package Of Asset Product By Now

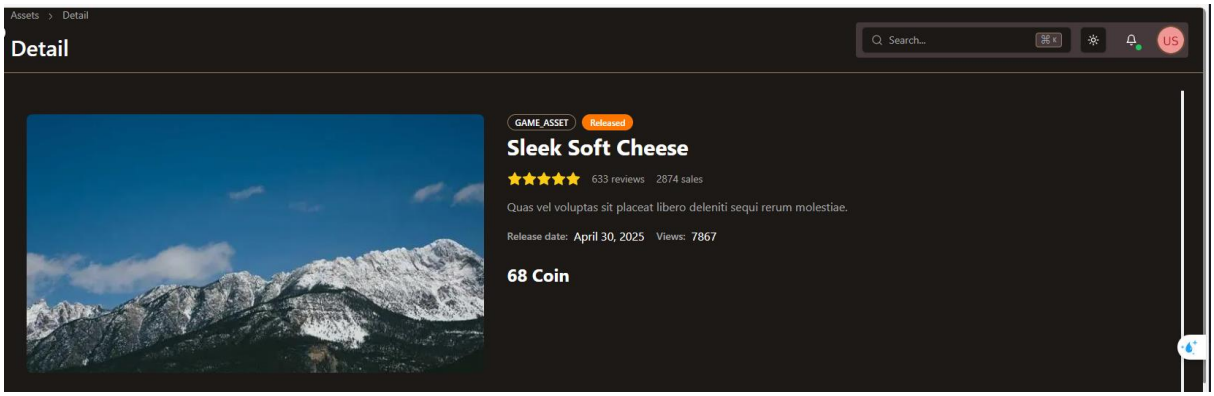

Figure 63 - Asset Package Showcase On Platform Store

3.11.7 Schedule Release Game Asset

Actor(s): Designer

- Function Trigger: Selects a game asset with status “Ready for Release” and clicks on “Schedule Release”.
- Function Description: The “Schedule Release Game Asset” function allows Designers to define a future date and time for releasing a game asset. This is useful for planning coordinated launches during events or for publishing updates without manual intervention at the time of release.
- Function Details: Steps to Schedule a Game Asset Release:
    1. Access the Game Asset
    2. Open Scheduling Form
    3. Configure Release Time
    4. Store Release Schedule
    - Post-Action Behavior:
        - Confirmation message: *“Asset release scheduled successfully.”*
        - The system automatically publishes the asset at the scheduled time and updates its status to Published.
    - Abnormal Cases:
        - Invalid date/time (e.g., past time): *“Please select a valid future release time.”*
        - System error while saving: *“Failed to schedule release. Please try again.”*

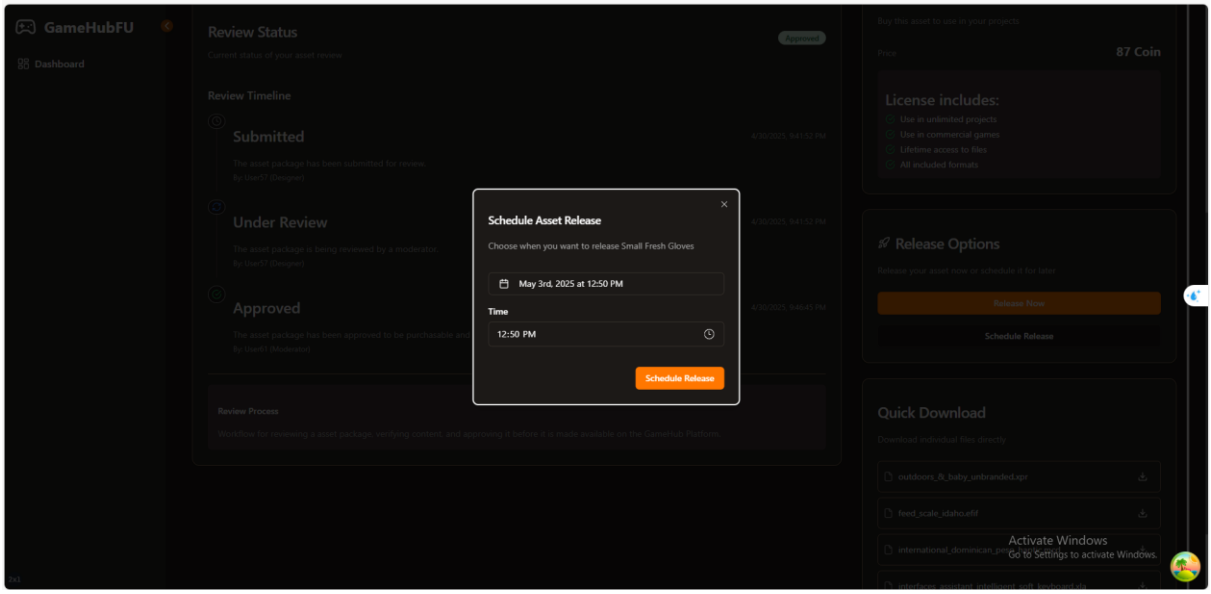

Figure 64 - Schedule Release Asset Package

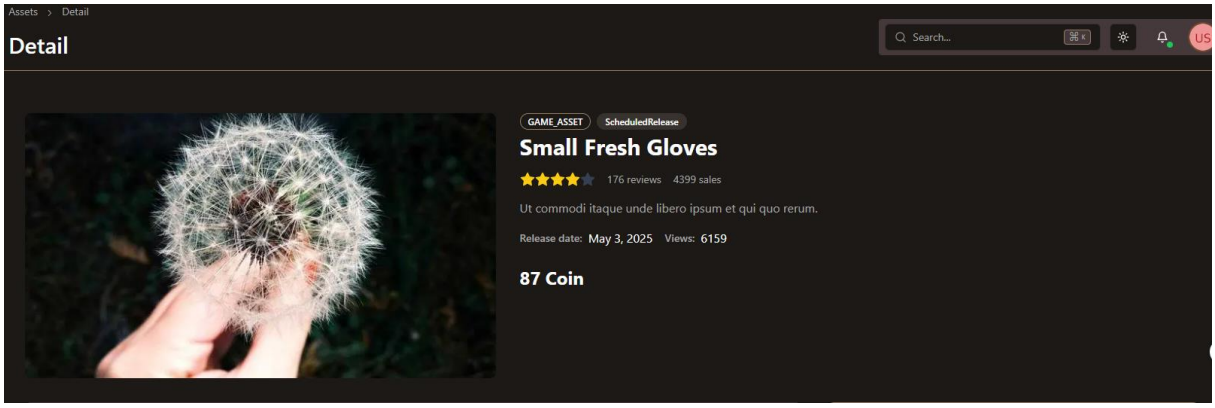

Figure 65 - Showcase Asset Package On Platform Store

3.12 Release Game Package, Game Asset Management

3.12.1 Send Request Get Release Approval For Game Package

Actor(s): Developer

- Function Trigger: The system shows the detailed status of the review request, including its current state, review history, and any actions that can be taken.
- Function: Function Description: The "Send Request Get Release Approval For Game Package" function allows the Developer to track and manage the status of their game package's review request. After submitting a game package for approval, the Developer can monitor its status, see if it's under review by a Moderator, and view any history of status changes. This functionality ensures that the Developer is informed about the progress of the package's review for publication.
- Function Details: Steps to Send Request for Release Approval:
    1. Submit the Game Package for Review
    2. Monitor Review Status
    3. Track Review History
    - Post-Action Behavior

- Once the package is submitted for review, the Developer must wait for the Moderator's decision.
- If the review is still under process, the Developer will see "Under Review" with a timestamp for when it was last updated.

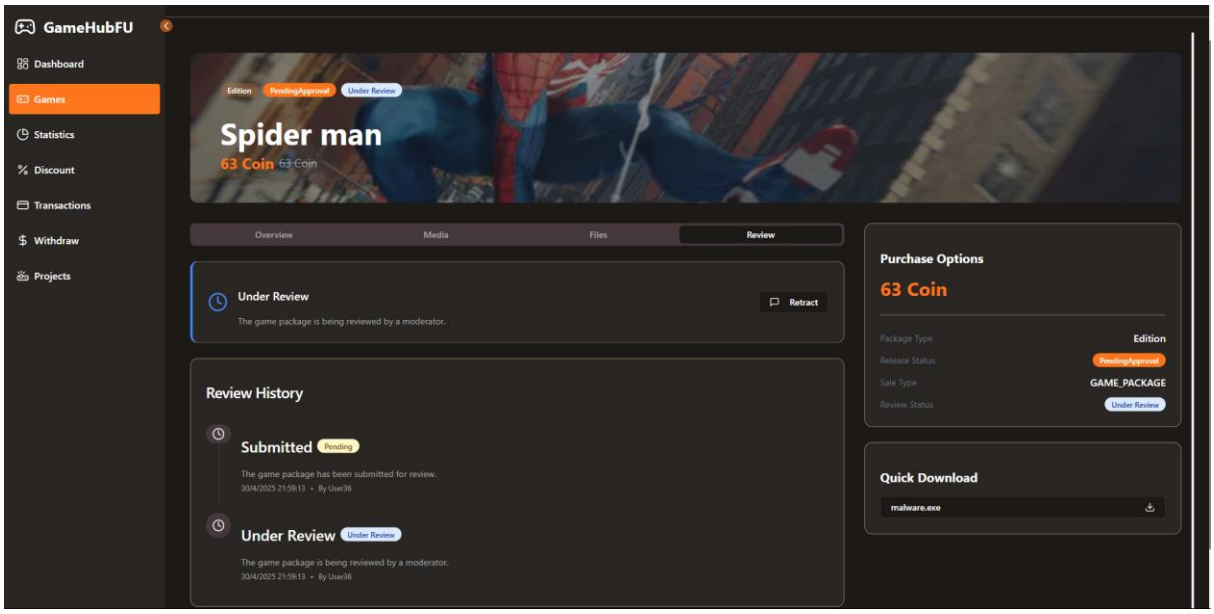

Figure 66 - Request Release Status Of Game Package

3.12.2 Send Request Get Release Approval For Game Asset

Actor(s): Designer

- Function Trigger: The Designer has filled in all required information such as description, pricing, license options, and file upload
- Function Description: This function allows the Designer to submit an asset for review by a moderator. The asset needs approval before it can be released for purchase or integration into projects. The Designer provides a description, licensing information, and other relevant details before sending it for review.
- Function Details: This function send the request release to moderator. The developer following the steps to submit asset for review:
    - Steps to Submit Asset for Review:
        1. Navigate to Asset Details
        2. Review the Asset Details
        3. Submit for Review
    - Post-Action Behavior:
        - If the asset is approved by the moderator, it will be released and available for purchase or use in projects.
        - If the asset is rejected, the Designer will be informed of the reasons for rejection and may make necessary changes.
        - The Designer can also schedule the release for later or choose to release it immediately.

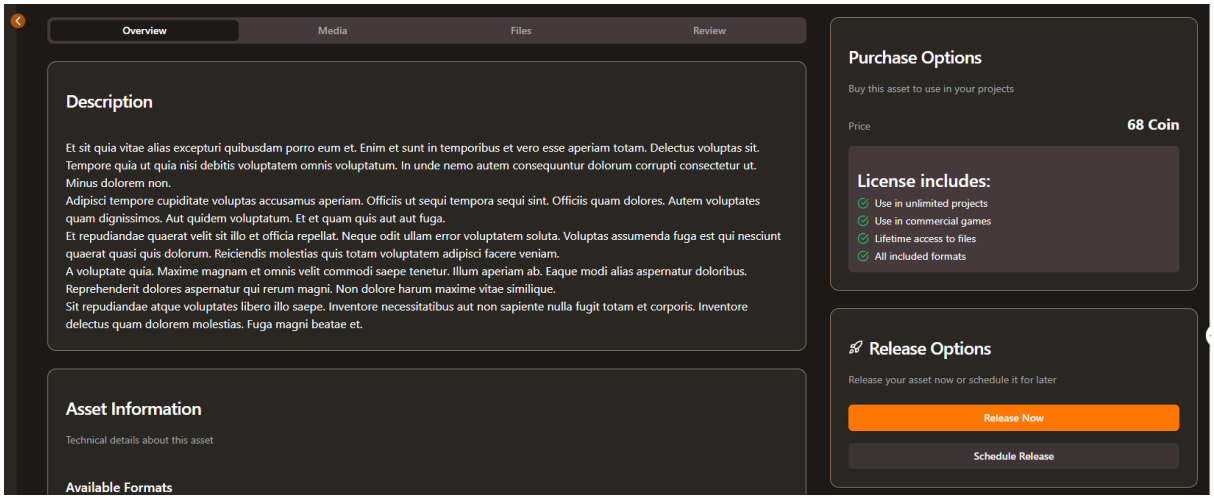

Figure 67 - Asset Send Request Review To Release Asset Package

3.12.3 Approve/Reject Submitted Release Requests

Actor(s): Moderator

- Function Trigger: The Moderator clicks on a game package that is under review in the "Game Package Reviews" section.
- Function Description: The Approve/Reject Submitted Release Requests function allows the Moderator to review and take action on game packages that have been submitted for review by Developers. The Moderator has three options for managing these requests: approve, reject, or require update from the Developer. This process ensures that game packages meet the platform's guidelines and are ready for publication.
- Function Details: This function show the current release request and the details information of package for reviewing before submit update status of request by following these steps:
    1. Access Review Request Details:
    2. Review Package Information:
    3. Approve the Package:
    4. Reject the Package:
    5. Request Update:
    - Post-Action Behavior:
        - If the "Approve" button is clicked, the game package is approved and prepared for release.
        - If the "Reject" button is clicked, the package is rejected, and the Developer receives feedback regarding the rejection.
        - If the "Require Update" button is clicked, the Developer is notified to update the game package, which will be reviewed again after the changes are made.

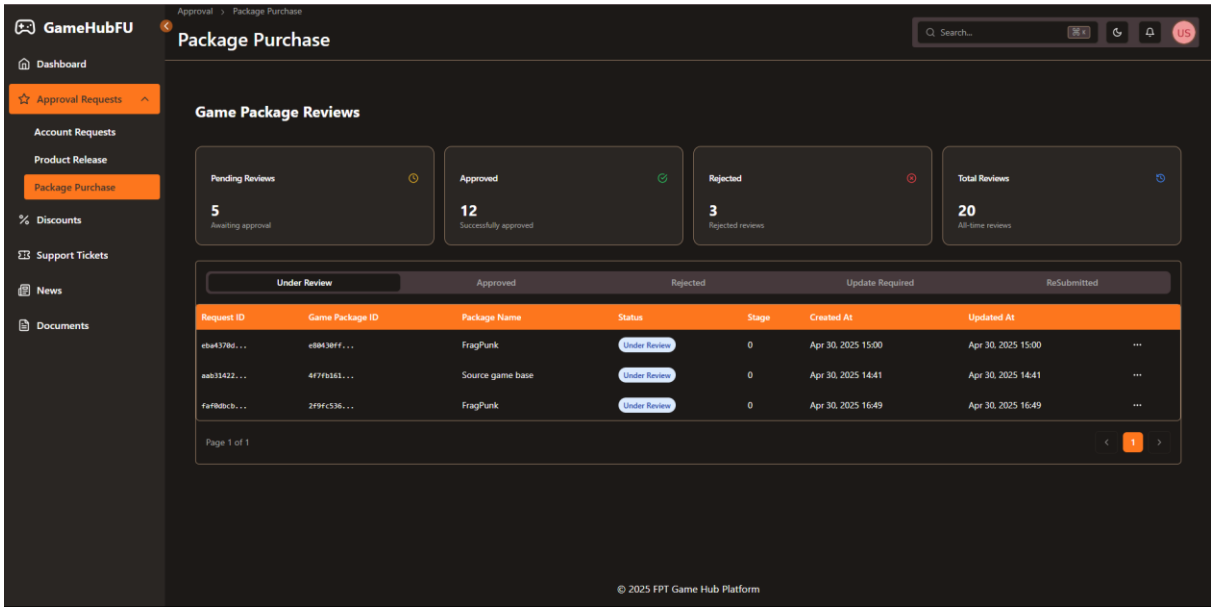

Figure 68 - View Under Review Request Release Packages

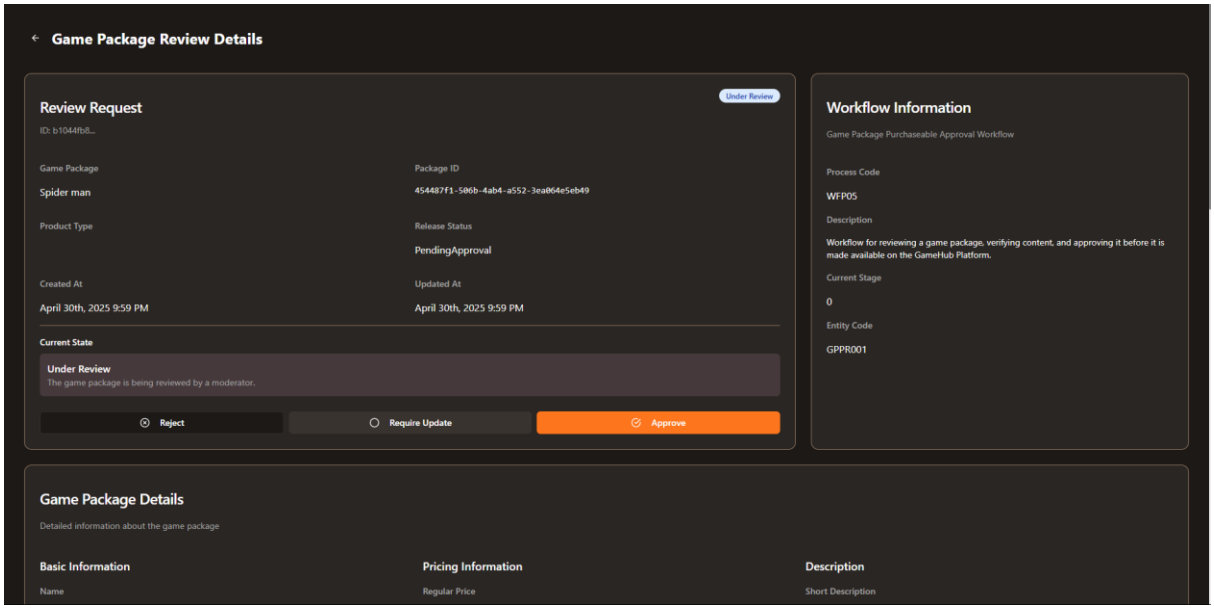

Figure 69 - Review The Request Release Package In Details

3.12.4 Track Submitted Release Requests

Actor(s): Moderator

- Function trigger: From the Dashboard or specific game/asset management page, each role will have access to the approval and review status of requests.
- Function Description**:** The "Track Submitted Release Requests" function allows different roles (Developer, Designer, and Moderator) to track and manage the release request status for game packages or assets. The status may include various stages like "Submitted", "Under Review", and "Approved". Each role has a specific view and functionality for managing the request based on their responsibilities.
- Function Details: This function show the list of submitted review requests with the status, the moderator do following actions:
    - Moderators can access a page to review submitted requests. They have the ability to approve, reject, or request updates.

- Once a request is under review, the Moderator can review the details and provide feedback or comments.
- The Moderator can take actions such as approving the request or requesting updates if there are issues.
  - Post-Action Behavior:
    - Once the package or asset is **approved**, it will be listed as "Approved" and will be available for purchase or use.
    - If it is **rejected**, the package will be marked as "Rejected" with comments, and the Developer/Designer will need to make necessary adjustments before resubmission.
    - If the submission is **under review**, the status remains "Under Review" until the Moderator makes a decision.

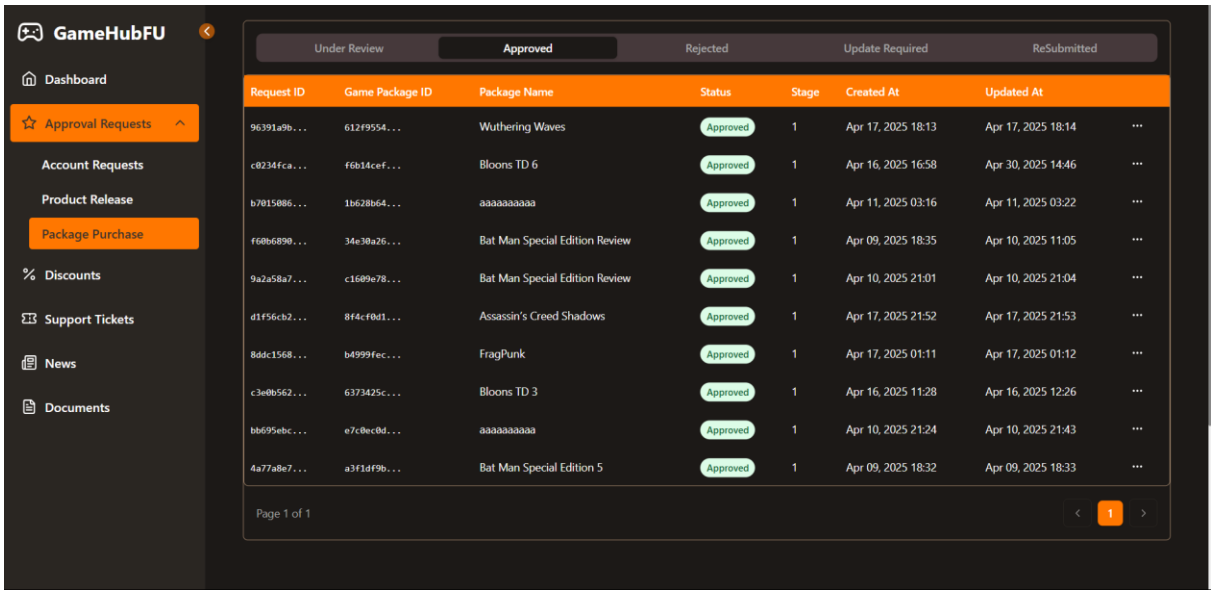

Figure 70 - View All The Request Release Package Base On Status Type

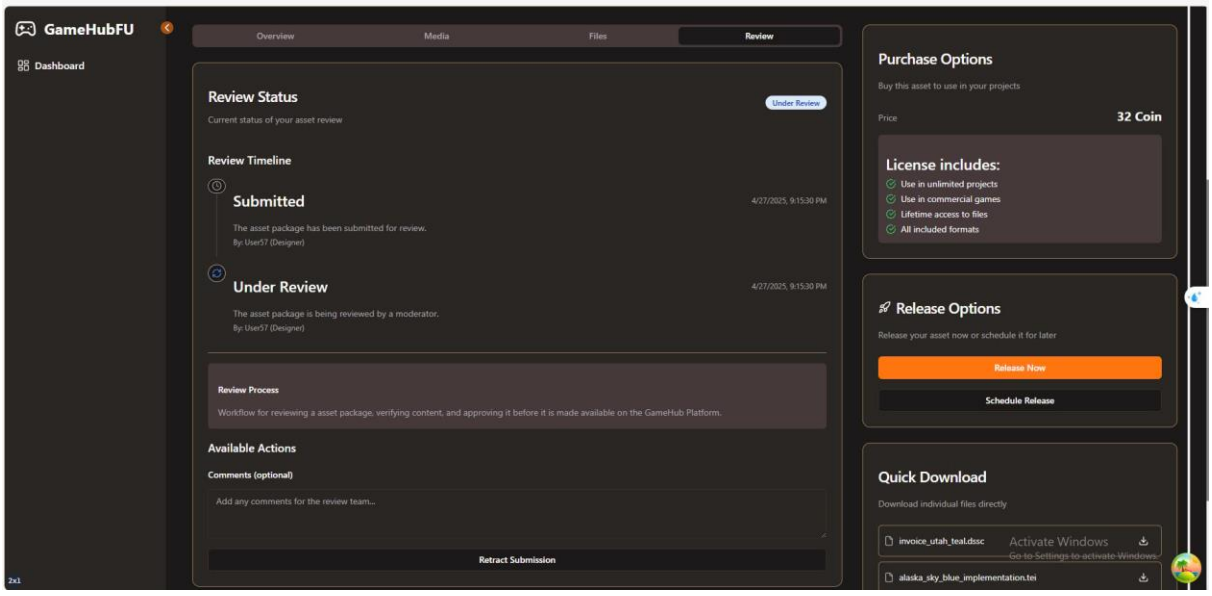

Figure 71 - Tracking the request review to release specific package

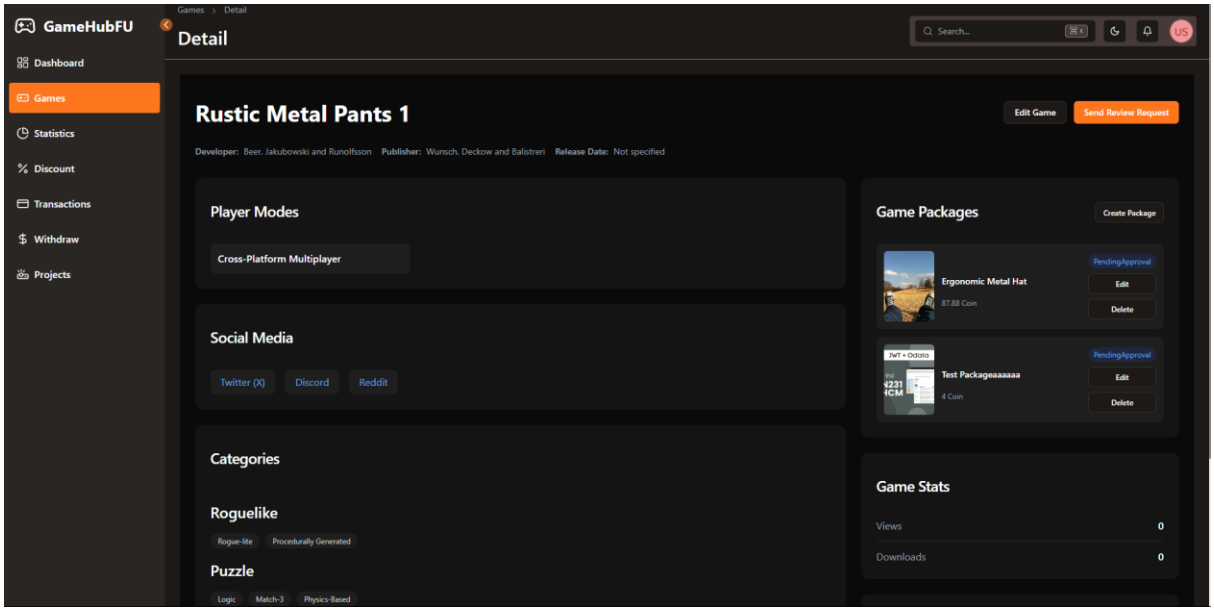

Figure 72 - View The Status Release Label Of Each Game Packages Of Game Product

3.13 Platform Discount Campaigns Management

3.13.1 Create New Platform Discount

Actor(s): Moderator

- Function trigger: The Moderator navigates to the “Platform Discounts” screen via the system sidebar and clicks the “+ Create Discount” button in the top-right corner.
- Function description: This feature enables Moderators to create new discount campaigns that apply platform-wide or selectively to specific packages on FPT Game Hub. These discounts are typically used for promotions such as seasonal events, flash sales, or global campaigns like Black Friday.
- Function details: This function allows the Moderator to define a discount campaign by filling out a form that includes the discount name, time range, percentage value, usage limit, cover image, and optional labels. The system supports global discounts that apply to all packages or scoped discounts that require manual assignment after creation. Once the Moderator submits the form, the system validates the inputs, stores the discount configuration, and marks the campaign as active during the specified timeframe. The discount data is then used by the platform to calculate reduced pricing on applicable items or packages.
    - Abnormal Cases:
        - Missing or invalid required fields : “Please complete all required fields correctly.”
        - System failure during creation: “Failed to create discount. Please try again.”
    - Success Case:
        - The system confirms successful creation with a message: “New discount created successfully.”
        - If the discount is not global (i.e., "Enable for All Packages" is off), the Moderator remains on the screen or is redirected to assign specific packages.
    - Precondition:

        - The user must be a Moderator or hold equivalent permissions to create platform-wide discounts. The platform must be operational and allow creation of promotional entries.
    - Postcondition:
        - A new discount campaign is stored and active within the defined date range. The discount is either applied globally or awaits package assignment depending on configuration.

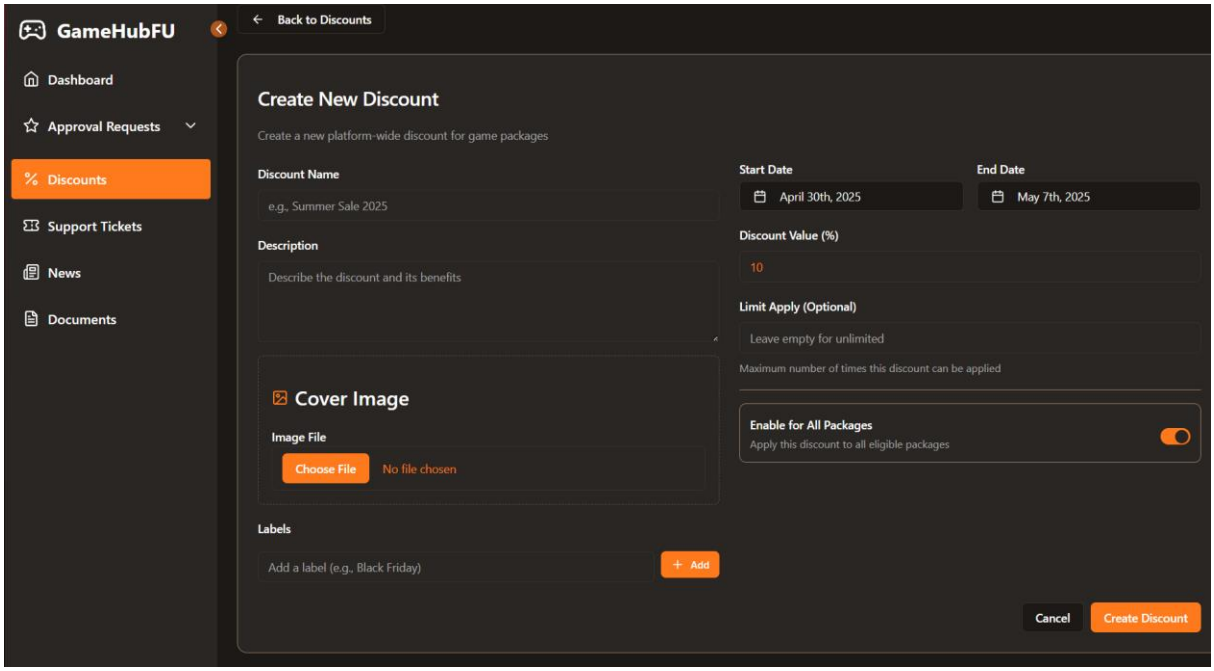

Figure 73 - Create New Discount

3.13.2 View All Game Platform Discounts In List

Actor(s): Moderator

- Function trigger: The Moderator navigates to the “Discounts” screen via the sidebar menu in the GameHubFU dashboard.
- Function description: This feature enables Moderators to view, search, and manage all existing discount campaigns across the GameHubFU platform. It provides full visibility over both active and inactive sales events, helping ensure centralized control over promotional activities.
- Function details: This function retrieves a list of all platform discount campaigns and presents them in a structured table format. Each row displays key discount metadata such as campaign name, labels, organiser, event type, current status (Published, Unpublished, or Expired), and an actions menu.
    - The system allows Moderators to filter or search for discounts using the name or tags via the search bar. Discounts are visually categorized by status through color-coded indicators. The actions menu on each row allows further operations such as editing, viewing details, assigning packages, or deleting the discount.
    - Only discounts with active or unpublished statuses can be edited or reassigned. Expired discounts are locked for editing unless duplicated for reuse.
    - Abnormal Cases:
        - No discounts available in the system: “No discounts found. Create one to get started!”

- Backend or fetch failure: “Failed to load discounts. Please refresh the page or try again later.”
- Success Case:
  - The system successfully loads all discount campaigns into a browsable and searchable table. Moderator can view, edit, or take further actions on each campaign as needed.
- Precondition:
  - Moderator must have permission to access the Discounts screen. At least one discount may exist, or the system supports viewing an empty list.
- Postcondition:
  - A list of all platform discount campaigns is displayed. Moderator can proceed to manage or create new discount entries using the provided interface.

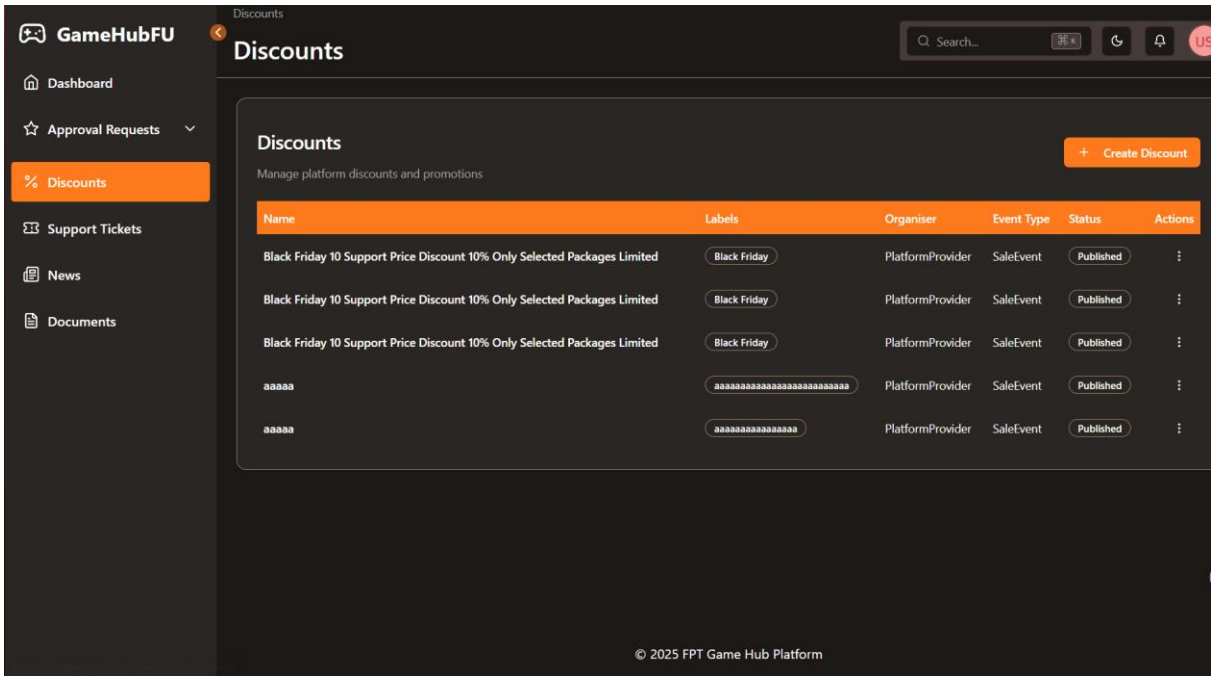

Figure 74 - Get Current Discounts List

3.13.3 View Game Discounts Information

Actor(s): Developer, Designer, Moderator

- Function trigger: The Developer or Designer navigates to the “Discounts” screen via the sidebar menu in the GameHubFU dashboard.
- Function description: This feature enables Developers and Designers to view all existing platform discount campaigns on the GameHubFU system. It supports visibility into both active and inactive discounts, allowing content teams to align product releases or updates with ongoing platform-wide promotions. Access is strictly read-only — modification functions such as creation, editing, or deletion are hidden for these roles.
- Function details: This function returns a list of all discount campaigns available on the platform.
  - The data is presented in a table format, where each row includes key information such as campaign name, category labels, organiser, event type, and current status.
  - Developers and Designers can search by discount name or label using the search bar. All discounts are color-coded based on status to improve visual recognition.
  - The “Create Discount” button and row-level actions (edit, delete, assign packages) are not displayed for these roles.

- Discount status definitions:
    - Published: Actively applied at checkout.
    - Unpublished: Not yet active or visible to users.
    - Expired: No longer valid due to elapsed end date.
- Abnormal Cases:
    - If no discount campaigns exist: “No discounts found. Contact Moderator to create one.”
    - If data retrieval fails: “Failed to load discounts. Please refresh the page or try again later.
- Success Case:
    - The system successfully displays all available discount campaigns in a searchable, scrollable table. Developers and Designers can browse campaigns to gain visibility into promotional strategies.
- Precondition:
    - Users must be logged in with a Developer or Designer role.
- Postcondition:
    - A full list of platform discounts is returned and displayed. The user can search and view but cannot perform any management actions.

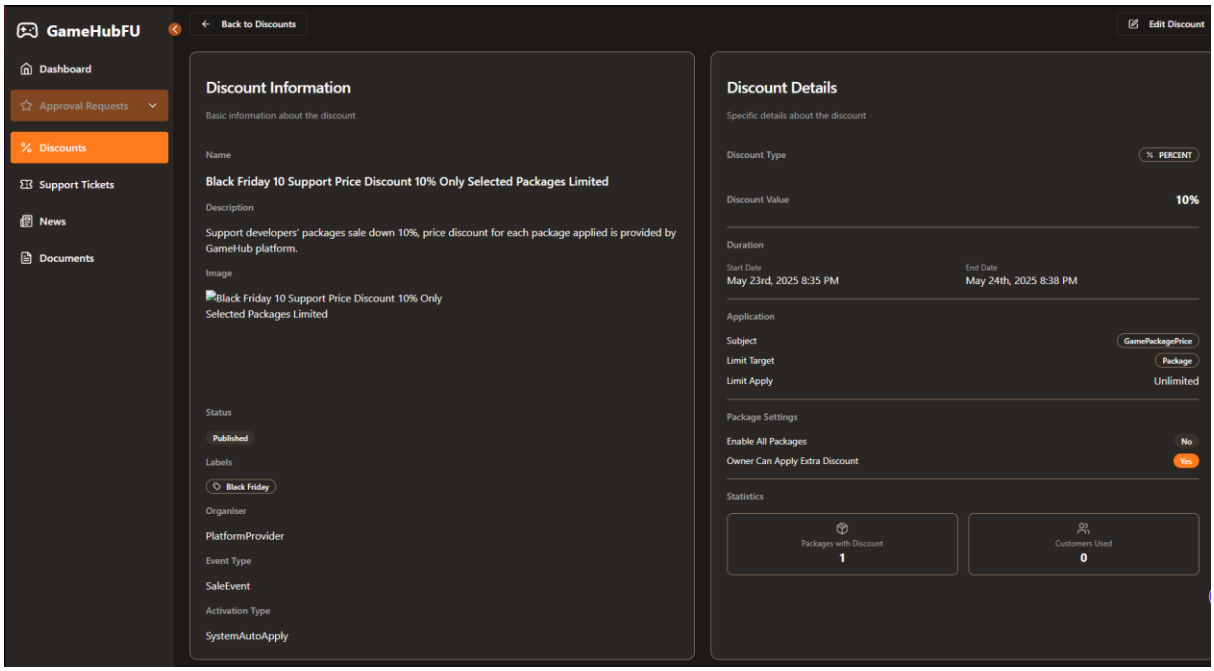

Figure 75 - View Discount Information In Details

3.16.4 Register Apply Platform Discount On Game Product

Actor(s): Developer

- Function trigger: The Developer navigates to the “Discounts” section via the sidebar menu in the GameHubFU dashboard, then selects a specific game package to view and register available discount campaigns.
- Function description: This feature enables Developers to view available discount campaigns that can be applied to their published game packages. It supports registering promotional offers for specific packages to help drive user engagement and increase sales during marketing campaigns.

- Function details: This function presents a list of the developer's game packages, each showing its current price and an option to manage discounts. Upon selecting a package, the system displays two tabs:
    - Available Discounts: Shows all discount campaigns currently available for the selected package.
    - Discount History: (Optional) Displays a history of past discounts applied to the package.
    - In the Available Discounts tab:
        - Each discount is listed with details such as name, tags, type, discount percentage, and status (e.g., Published).
        - Each row includes an Actions dropdown with options like View Details and Register.
        - Developers can use filters and search to find relevant discounts by name, type, status, or creation date.
    - Only discounts that meet package eligibility criteria and are in "Published" state can be registered. When a developer clicks Register, the system attempts to apply the discount to the selected package, updating its price if successful.

    - Abnormal Cases:
        - Package already has an active discount: "This package already has an active discount. Please wait until it expires before registering a new one."
        - Discount is expired or inactive: "This discount is no longer valid."
        - Package is not eligible (e.g., unpublished or under review): "Only published packages can be registered for discounts."
        - Server error during registration: "Something went wrong while registering the discount. Please try again later."
    - Success Case:
        - The system successfully registers the selected discount to the game package. The package price is updated accordingly, and the discount appears in the discount history (if applicable).
    - Precondition:
        - The Developer is authenticated and has at least one published game package.
        - At least one active and applicable discount exists on the system.
    - Postcondition:
        - The selected package is registered to the discount, and its display price is updated.
        - The developer may proceed to track the impact of the discount or manage future promotions.

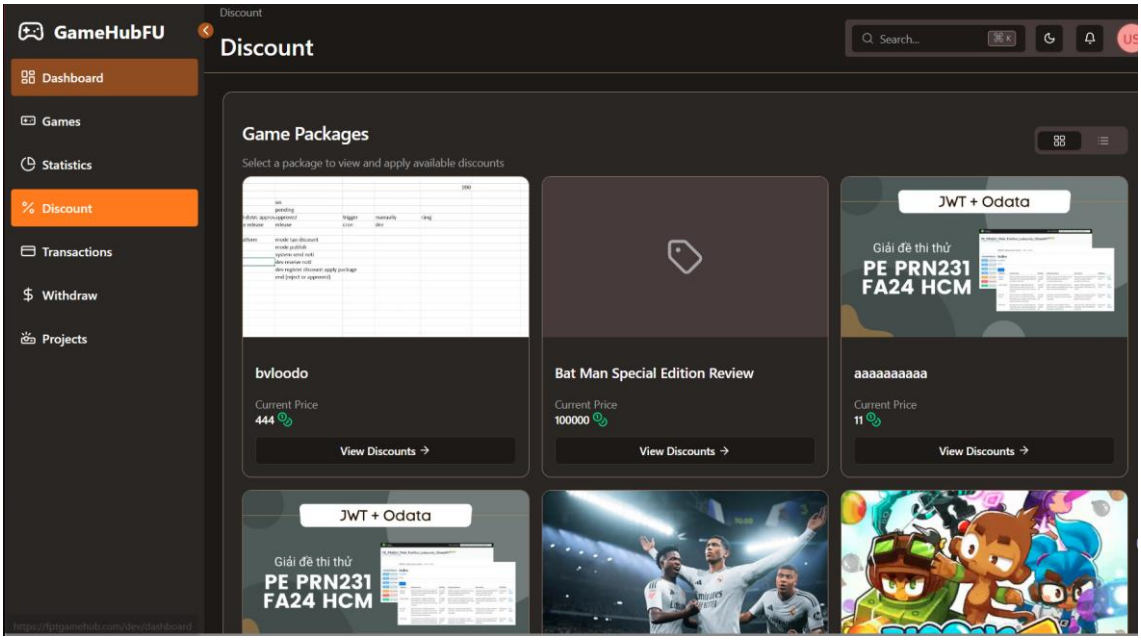

Figure 76 - Get Packages Can Apply Discounts

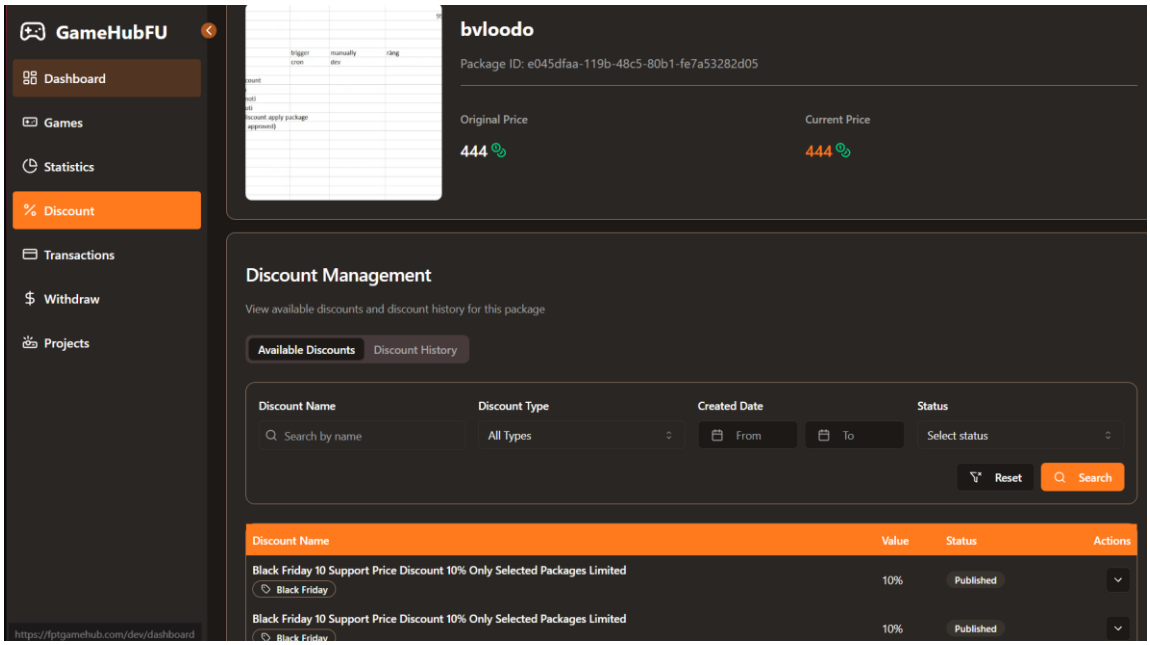

Figure 77 - View Package And Its Discounts In Details

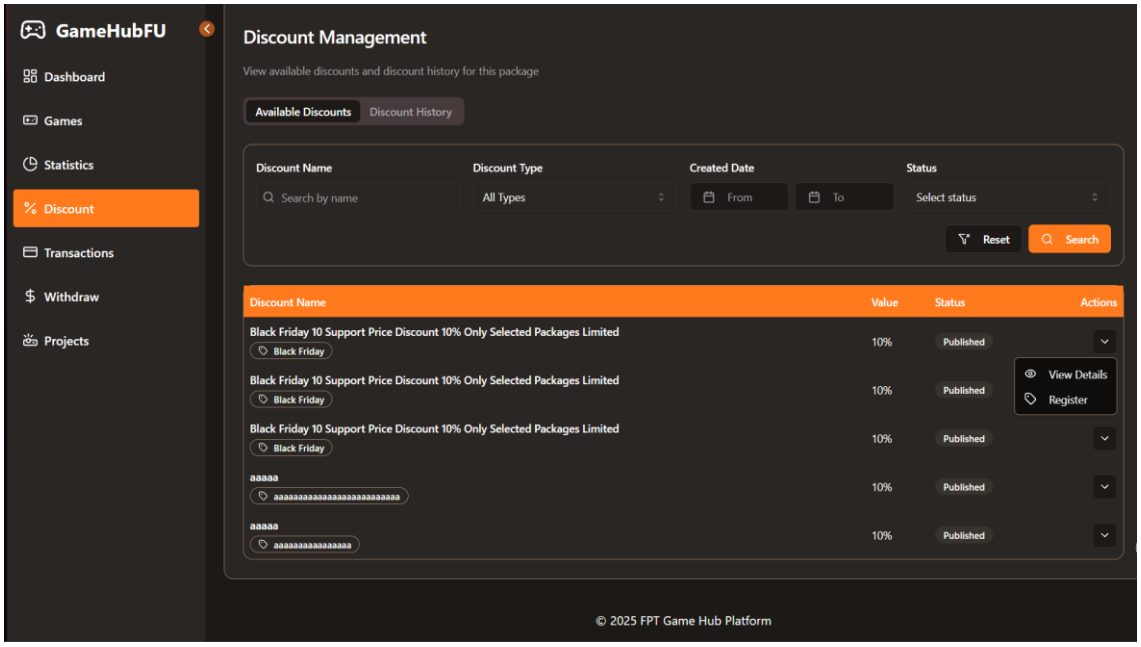

Figure 78 - Select One Discount To Register Apply

3.14 Browse Game Package, Asset Package

3.14.1 Search, Sort, Filter On Released Game Products On Store

- Function Trigger: The user enters a search keyword or selects filters and clicks "Apply"
- Function Description: The Game Search Page allows users to search, sort, and filter released games in the store. Users can apply filters such as price range, minimum rating, release date, and game genre.Results will display information such as the game title, price, release date, and user rating.
- Function Details: Displays detailed information about the game, including description, price, release date, rating, and options to purchase the game.

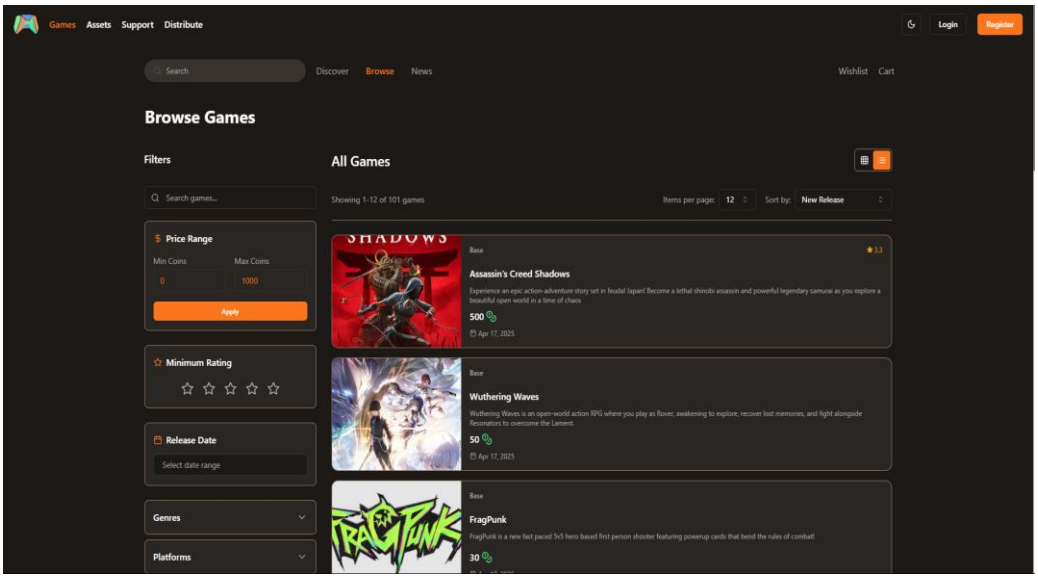

Figure 79 - Search, Sort, Filter Game and Asset Packages On Store

3.14.2 View Released Game Packages On Store In Details

Actor(s): Customer

- Function trigger: User clicks on the game in the store's browse section to view more details.
- Function description: The Game Detail Page allows users to view detailed information about a released game product in the store. It includes information such as the game's title, description, ratings, price, and release date. The page also provides options for adding the game to the cart or wishlist. Additional game statistics like views and downloads are displayed. A video player is embedded, allowing users to view a trailer or gameplay footage, along with additional images showcasing the game.
- Function details: The function displays detailed information about the selected game, including:
    - Game Title: The name of the game.
    - Description: A brief summary of the game's story and features.
    - Rating: Average user rating of the game.
    - Price: The cost of the game in coins.
    - Release Date: The official release date of the game.
    - Add to Cart/Wishlist: Buttons for adding the game to the cart or wishlist.

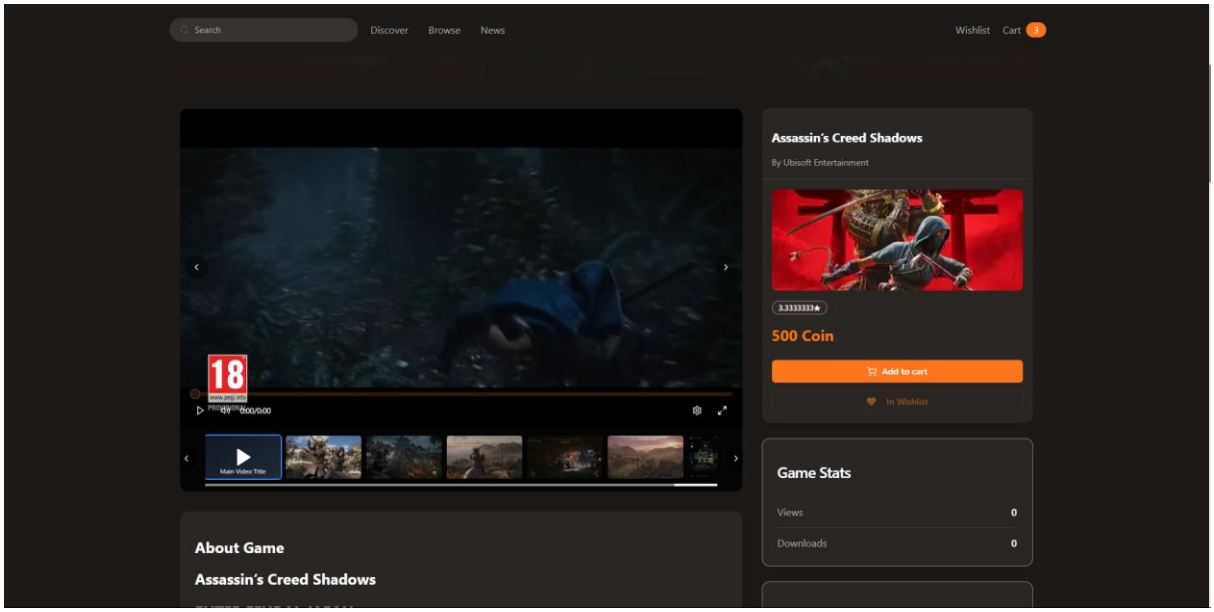

Figure 80 - View Package In Details On Platform Store

3.14.3 Search, Sort, Filter On Released Game Assets On Store

Actor(s): Customer

- Function trigger: Displays the game assets that match the selected criteria, sorted or filtered accordingly. The results will be updated immediately when the user changes the criteria.
- Function description: The Game Asset Search Page allows users to search, sort, and filter released game assets in the store. Users can apply filters such as price range, minimum rating, release date, and asset type. The search results will display information such as the asset title, price, release date, and user rating.
- Function details: Displays the game assets that match the selected criteria, sorted or filtered accordingly. The results will be updated immediately when the user changes the criteria.

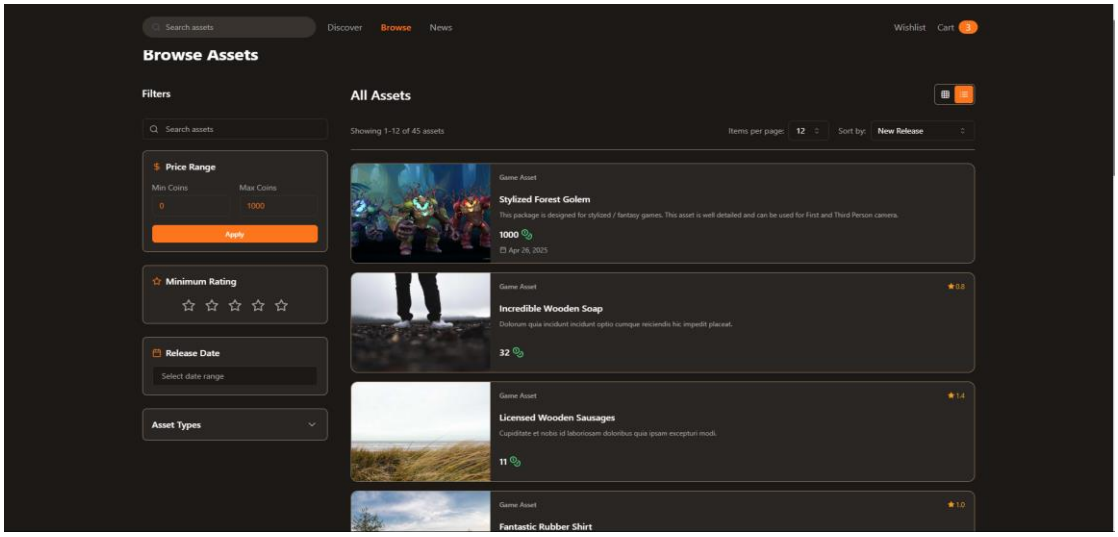

Figure 81 - Search, Sort, Filter Asset On Platform Store

3.14.4 View Released Game Asset On Store In Details

Actor(s): Customer

- Function trigger: The user clicks on a game in the store to view details.
- Function description: The Game Detail Page allows users to view detailed information about a released game in the store. Users can view information such as the game title, price, release date, description, and user ratings. If the game is not purchased, the "Add to Cart" button will appear. If the game has already been purchased, the "View in Library" button will allow users to access their library. Additionally, users can view support contact information such as email, phone number, and the developer's website.
- Function details: Displays detailed information about the game, including:
    - Asset Title: The full name of the asset.
    - Price: The price of the asset in Coin.
    - Release Date: The official release date of the asset .
    - Description: A detailed introduction to the game.
    - Rating: The average user rating of the asset.

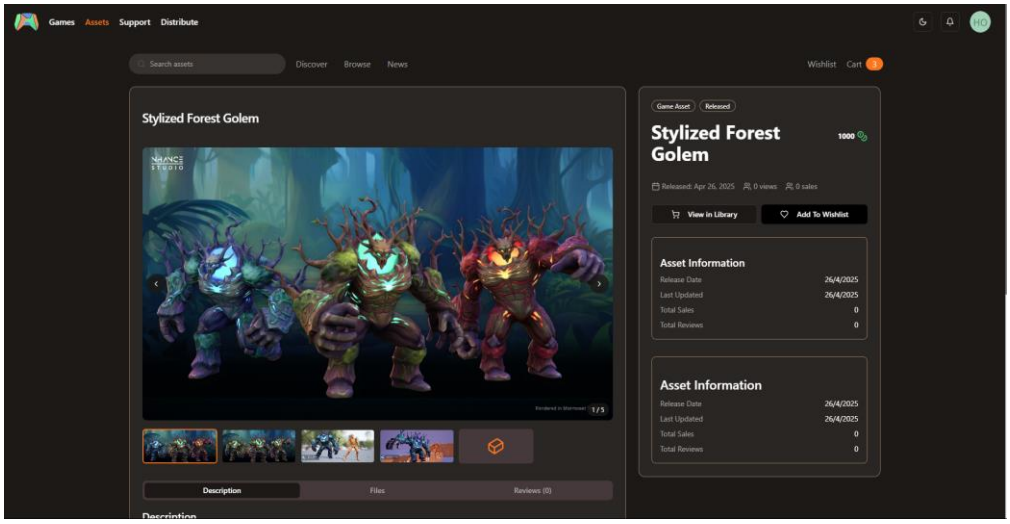

Figure 82 - View Asset Package Details On Platform Store

3.14.5 Save Game Product, Game Assets In Wishlist

Actor(s): Customer

- Function trigger: The "Create New" button to create a new wishlist, located within the wishlist creation window.
- Function description: Users can create a wishlist folder to store their favorite game products or game assets. When a user wants to add a game or game asset to the wishlist, they simply click the "Add to Wishlist" button. After the game or game asset is added, the "Add to Wishlist" button will change to "View in Wishlist". Users can access their wishlist to view and manage the games they have saved.
- Function details: The function requires the user enters the name and description for the new wishlist. After creating, the new wishlist will appear in the list for the user to add games to.

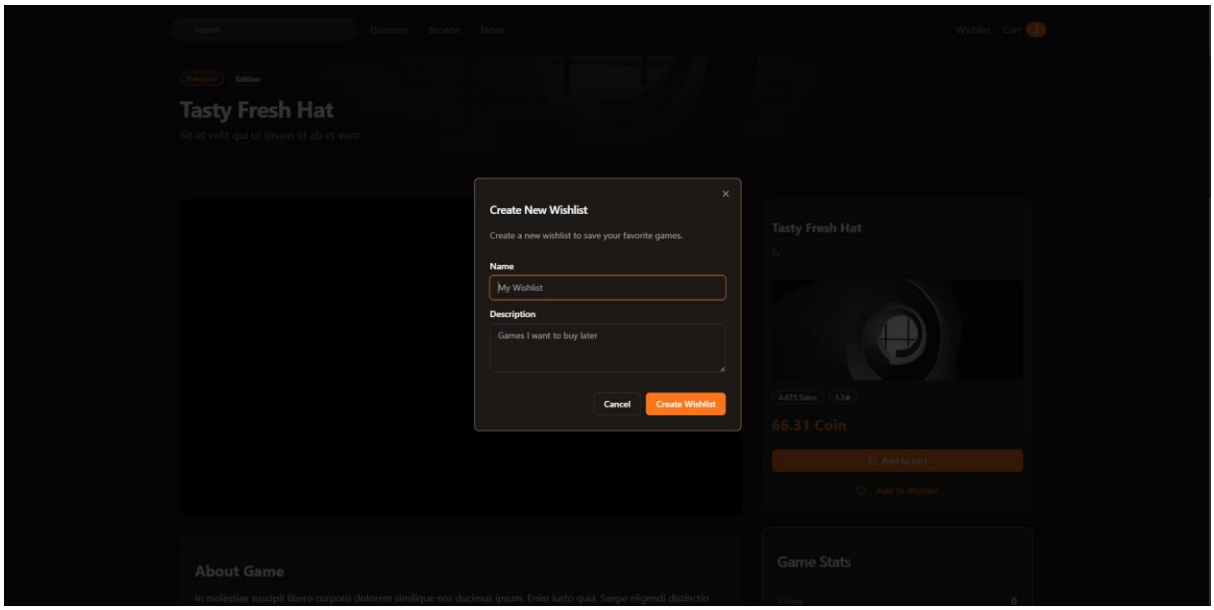

Figure 83 - Add New Wishlist

3.14.6 Purchase Game Product, Game Assets By Coins

- Actor(s): Customer
- Function Trigger: The user clicks the **"Checkout"** button to proceed to the checkout page.
- Function Description: The Cart page displays the game products or game assets that the user has selected. The user can view a summary of the order, including the price, any sale discounts, and the total amount to be paid. When the user clicks the "Checkout" button, the checkout page will display detailed information about the order. After the user successfully completes the payment, a success notification will appear, and the user will be redirected to the Library page to view the purchased items.
- Function Details: The function displays the product information, total amount to be paid, and the user's coin balance. After the payment is successful, the "Buy" button changes to a success message, and the system automatically redirects the user to the Library page.

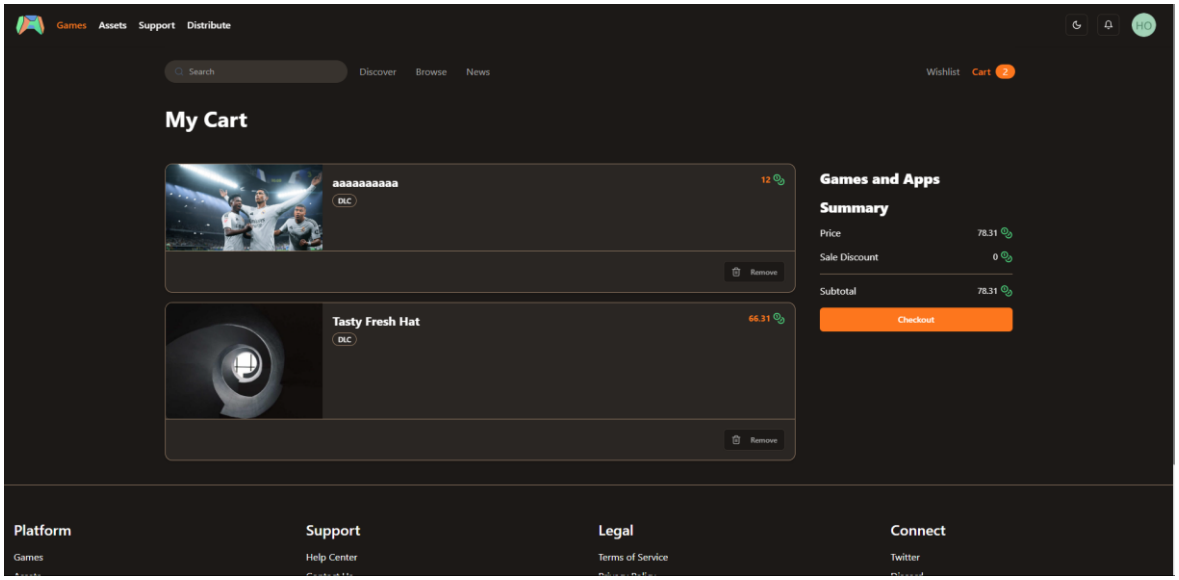

Figure 84 - View Account Cart

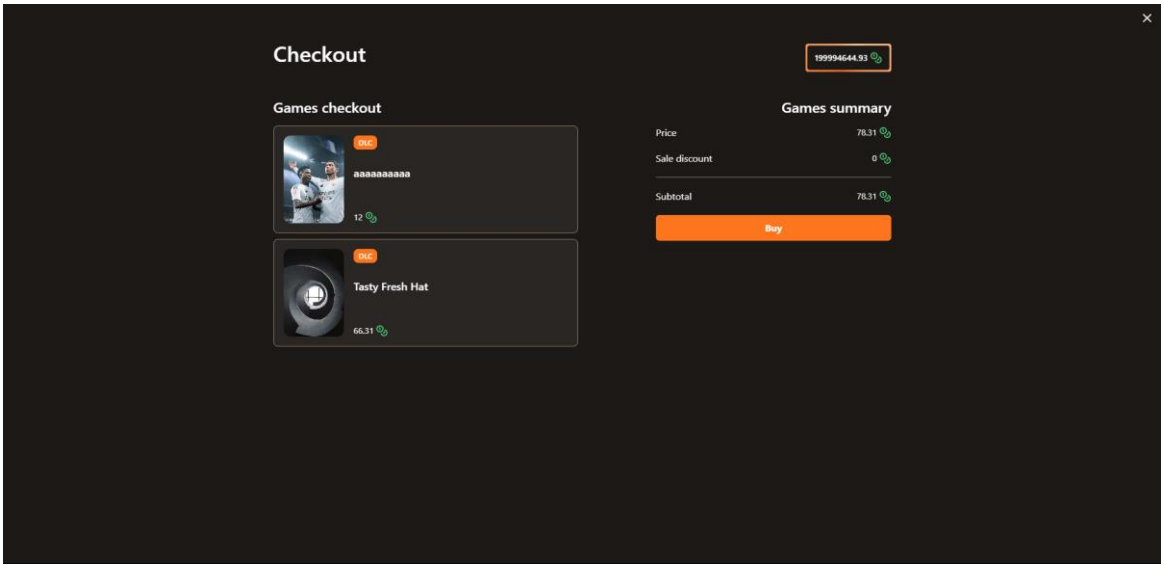

Figure 85 - Checkout Buy Products

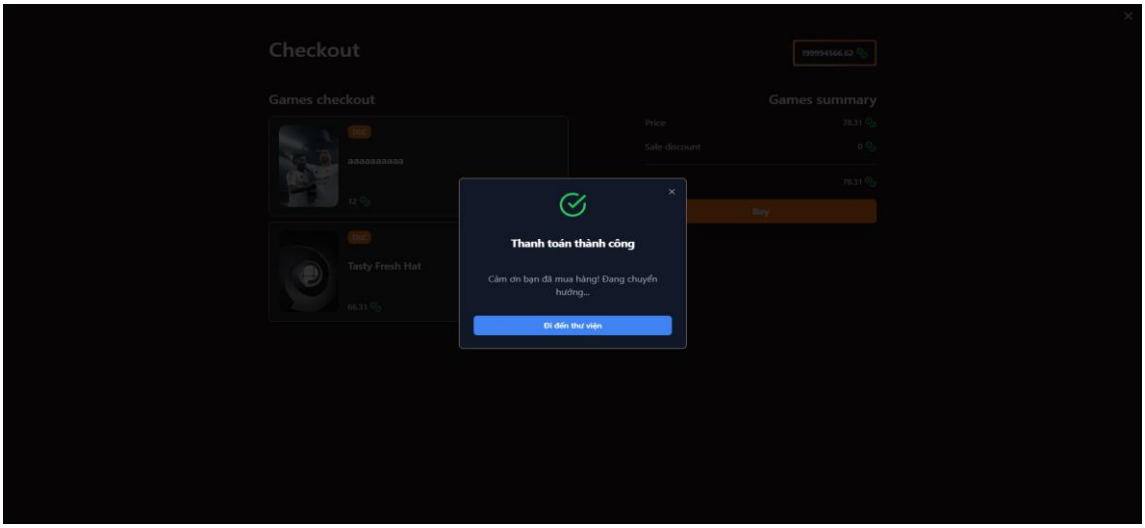

Figure 86 - Buy Products Successfully

3.14.7 Download Executable Game Of Purchased Game Products

Actor(s): Customer

- Function trigger: The Customer clicks the "Download" button next to the game product.
- Function description: The "Download Executable Game Of Purchased Game Products" function allows the Customer to download the executable file of a purchased game product. After purchasing a game, the customer can directly access and download the game to their system to begin playing.
- **Function details: The functions download executable game by following steps:**
    1. Navigate to My Game Library:
    2. Select the Game to Download
    3. Click the "Download" Button
    4. Complete the Download
    - Post-Action Behavior:
        - Once the download completes, the Customer can launch the game from their device.

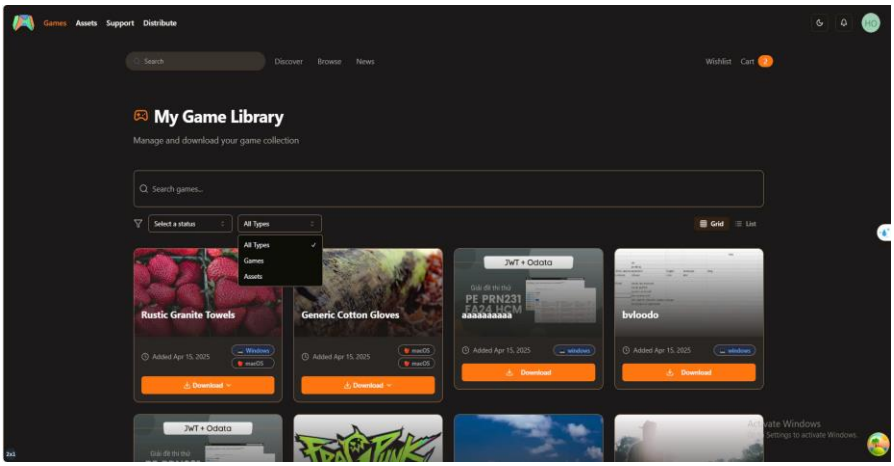

Figure 87 - View Purchased Digital Products In Library

3.14.8 Download Files Asset Of Purchased Game Assets

- Actor(s): Customer
- Function trigger: The Customer clicks the "Download" button next to the asset they wish to download.
- Function description: The "Download Files Asset Of Purchased Game Assets" function allows the Customer to download the files of game assets they have purchased. This includes various files (e.g., models, textures, scripts) that the customer can use in their game development or creative projects.
- Function details: Steps to Download Game Asset Files
    1. Navigate to My Asset Library
    2. Select the Asset to Download
    3. Click the "Download" Button
    4. Complete the Download
    - Post-Action Behavior:
        - After the download completes, the Customer can use the asset files in their projects or games.

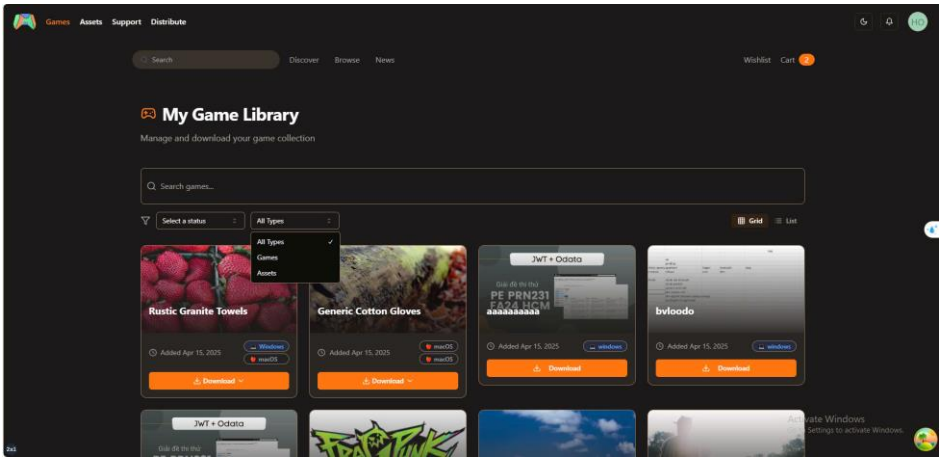

Figure 88- Select Digital Type To View Purchased Asset Products

3.15 Feedback On Purchased Game Product, Game Asset

3.15.1 Send Feedback On Purchased Game Product/Game Asset

Actor(s): Customer

- Function trigger: Player submits a review or rating for a previously purchased game product or game asset.
- Function description:This feature allows a player to submit feedback for a purchased game asset or game product. The feedback includes a rating, recommendation flag, and an optional comment. The submitted feedback helps developers and other players assess the quality of game packages.
- Function details: The system sends rating and comment comment messages to the user.
    - Abnormal Cases:
        - Invalid or missing gamePackageId: API returns 400. Display message: “Unable to submit feedback. Please try again.”
        - Feedback for non purchased product: API returns 403 Forbidden. Show: “You can only review assets you’ve purchased.”
        - Invalid rating value (e.g., negative, too high): API returns validation error. Prompt user to enter a valid score.
        - Duplicate submission: API may reject multiple feedbacks for the same package. Display: “You’ve already submitted feedback for this item.”
    - Success Case:
        - Feedback is successfully submitted and stored. The system may return a success message or confirmation. The feedback is then available for display in the asset’s detail page or review section.
    - Precondition:
        - The player is authenticated.
        - The player has purchased the game product or asset.
        - A valid gamePackageId is provided.
    - Postcondition:
        - Feedback is stored and becomes part of the asset’s review data.
        - Developers can retrieve and analyze user feedback.
        - Other players may view the rating and comments in the UI.

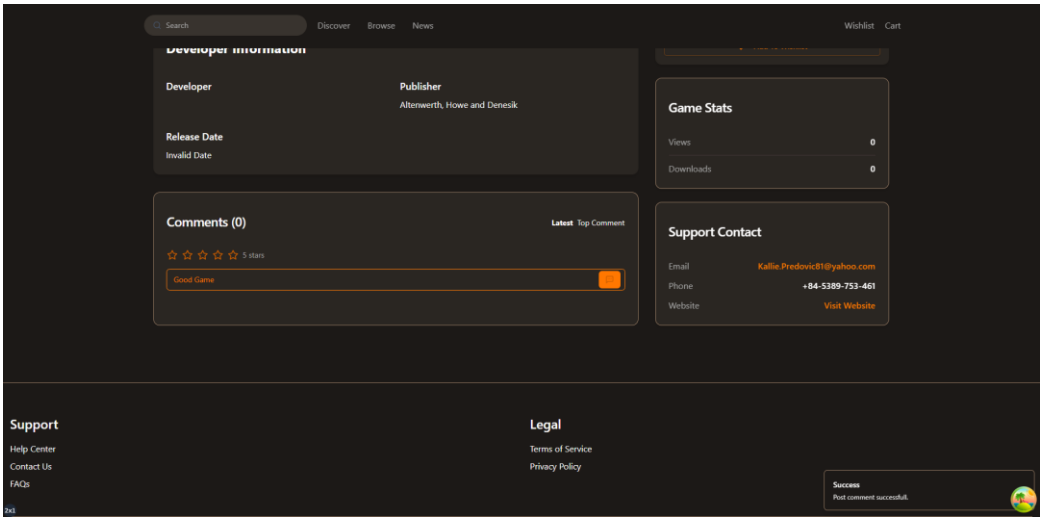

Figure 89 - Send Feedback To Package

3.15.2 View All Feedbacks Of Specific Game Product/GameAsset In List

Actor(s): Customer

- Function trigger: A user navigates to a game product or game asset detail page where feedback is displayed.
- Function description: This feature enables the system to retrieve and display all submitted feedback for a specific game product or asset. It helps players make informed decisions and allows designers or administrators to evaluate product performance and user satisfaction.
- Function details: This function retrieve feedback associated with the specified game package.
    - Abnormal Cases:
        - Missing or invalid gamePackageId: The API returns a 400 or 404. Display: “Unable to load reviews for this asset.”
        - No feedback submitted yet: API returns an empty array. Show: “No reviews yet. Be the first to share your experience!”
    - Success Case:
        - All valid feedback entries are successfully fetched and displayed in the UI, optionally sorted by most recent or highest rating. Users can browse through reviews and get insights before interacting with or purchasing the product.
        - Precondition:
        - A valid gamePackageId must be provided
        - The game package must exist in the system
        - Feedback entries must have been previously submitted
    - Postcondition:
        - Users can read all available feedback associated with the product.
        - User can use this data for quality improvement and analytics

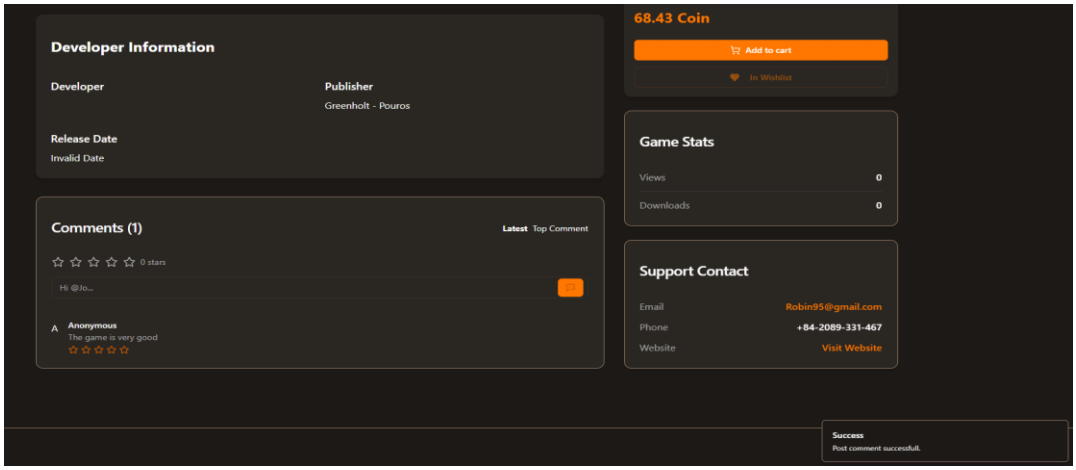

Figure 90 - View Feedbacks Of Package

3.16 Withdraw Coins Into Money

3.16.1 Send Withdrawal Request

Actor(s): Developer, Designer

- Function trigger: Developer or Designer logs into the Game Hub Platform, navigates to the Withdraw section, and initiates a new withdrawal request from their available earnings.
- Function description: This feature allows Developers and Designers to submit a withdrawal request to transfer their available Game Hub earnings to a registered bank account. The platform collects necessary banking information and verifies the withdrawal amount before processing the request. It ensures that users can manage and claim their earned revenue conveniently within the Game Hub ecosystem.
- Function details: This function allows Developers or Designers to create a withdrawal request by submitting the amount, bank name, account number, and account holder name, after which the system validates the data and marks the request as pending if all conditions are met.
    - Business Rules:
        - Minimum withdrawal amount is 100 coins.
        - Withdrawal amount must not exceed available balance.
        - Only verified Developer or Designer accounts are eligible.
        - Bank Name must be selected from the platform-provided list.
        - All banking information must be valid and complete before submission.
    - Abnormal cases:
        - Withdrawal amount min is 100 coins: "Minimum withdrawal amount is 100 coins."
        - Required fields missing : "Please fill in all required fields."
        - Amount exceeds available balance: "Insufficient available balance."
        - Submission error: "Failed to submit withdrawal. Please try again later."
    - Success case:
        - System validates all inputs and confirms the request: "Withdrawal request submitted successfully!"
        - The user is redirected to the Request History tab to track status.

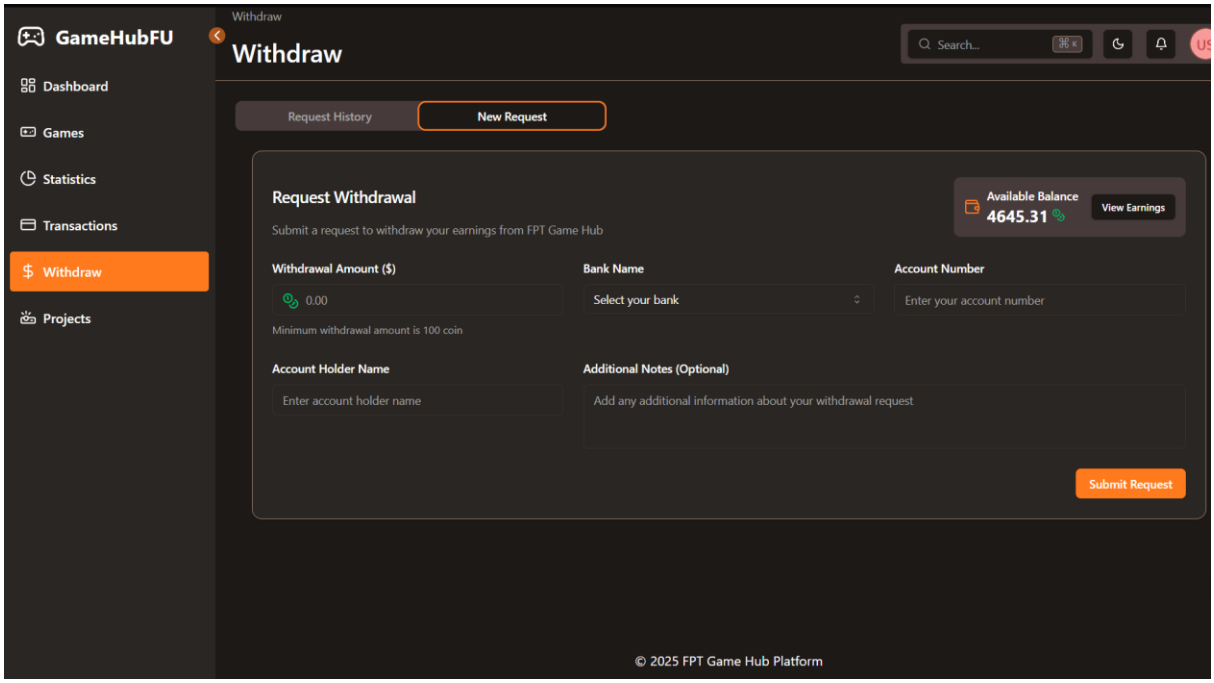

Figure 91 - Send Withdrawal Request

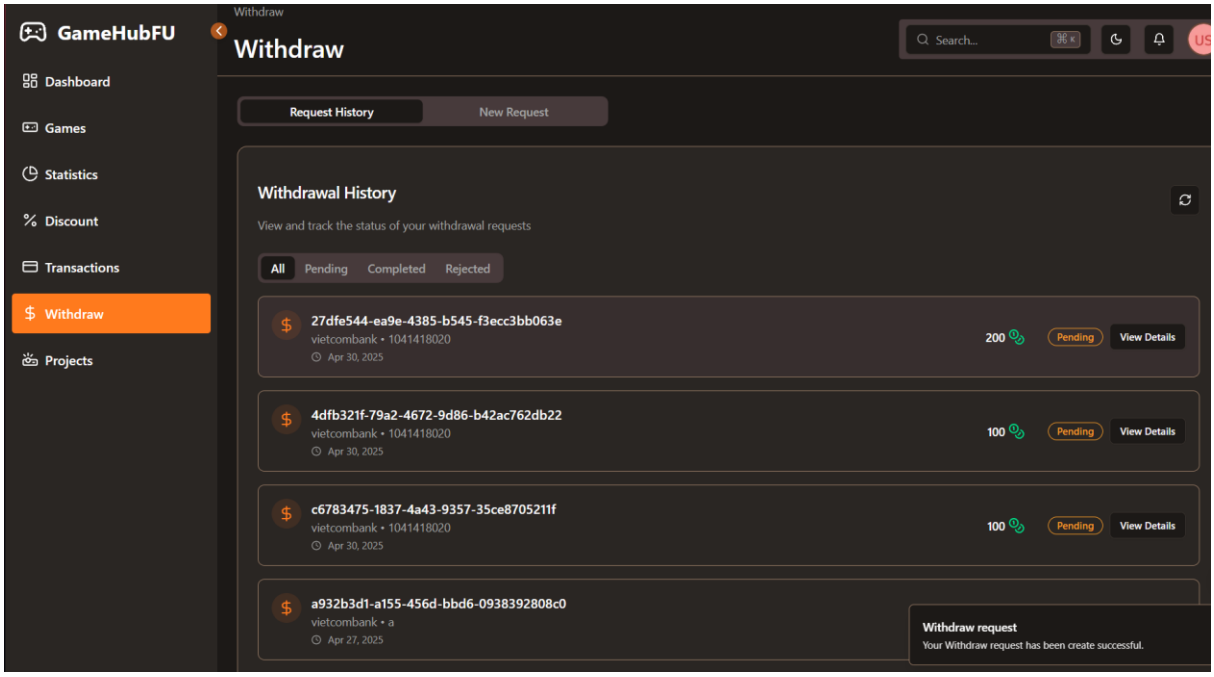

Figure 92 - Send Withdrawal Requests History

3.16.2 Approve or Reject Withdrawal Request

Actor(s): Admin

- Function trigger: Admin logs into the Game Hub Platform, navigates to the Withdrawal Request management section, selects a pending withdrawal request, and chooses to either approve or reject the request after verifying the transaction.
- Function description: This feature allows Admins to manage and process withdrawal requests submitted by Developers or Designers. Admins are responsible for verifying the

payment status and ensuring that funds have been successfully transferred to the user's registered bank account. This function ensures transparency and security in handling user withdrawals within the Game Hub ecosystem.

- Function details: This function enables Admins to approve or reject a pending withdrawal request. To approve, the Admin must upload a valid payment receipt. To reject, a rejection reason must be provided. Upon action, the system updates the request status accordingly and notifies the requester.
    - Business Rules:
        - Only requests in Pending status can be processed.
        - Approving a request requires uploading a valid image file as payment receipt.
        - Rejecting a request requires entering a non-empty rejection reason.
        - Request status is updated to "Approved" or "Rejected" based on action taken.
        - Developers/Designers receive a notification and can view the updated status.
    - Abnormal cases:
        - No payment receipt uploaded (when approving): "Please upload a payment receipt to approve the request."
        - No reject reason provided (when rejecting): "Reject Reason is required."
        - Invalid file format uploaded: "Invalid file format. Please upload a valid image."
        - Unauthorized access: "Only Admin users can process withdrawal requests."
    - Success case:
        - Admin uploads a valid receipt or enters a rejection reason: "Withdrawal request approved successfully!" or "Withdrawal request rejected successfully!"
        - The system updates the status immediately and logs the action in the request history.

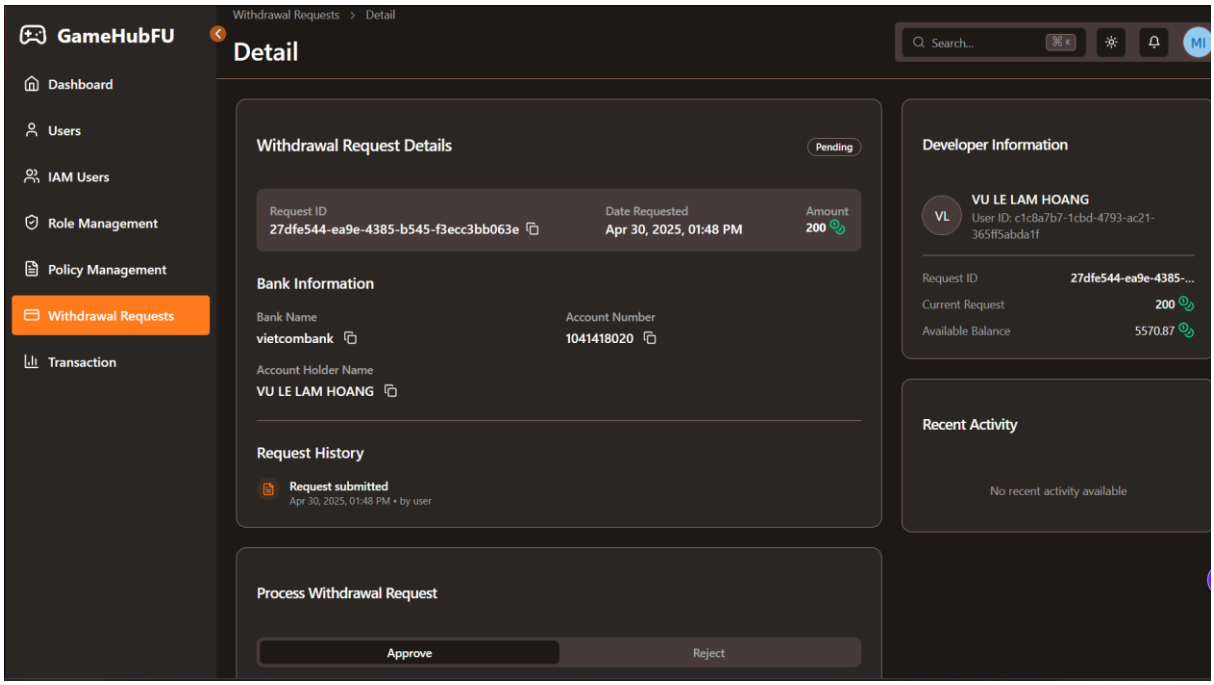

Figure 93 - Admin View Withrawal Requests

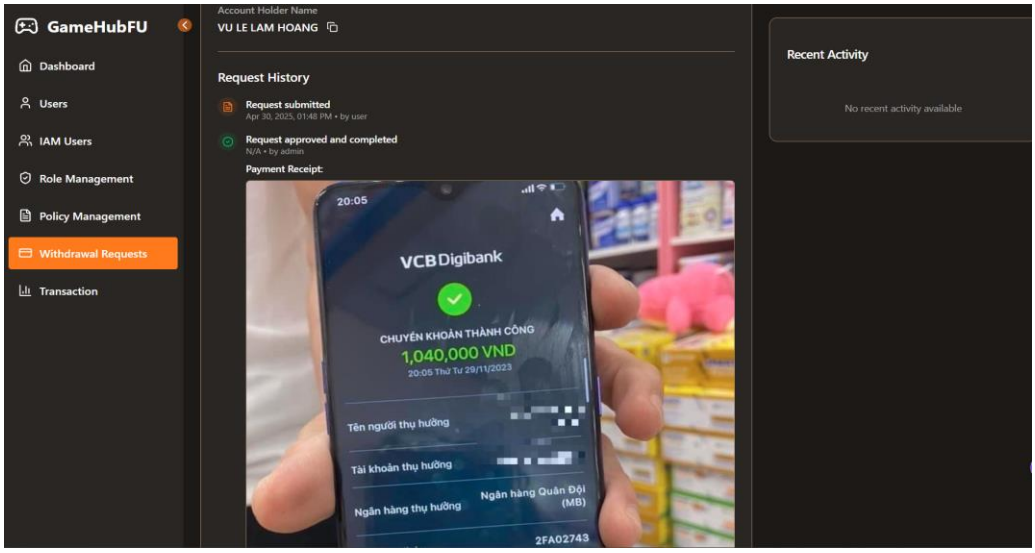

Figure 94 - Admin Approve Withdrawal Request With Evidence Send Money Successfully

3.16.3 View All Submitted Withdrawal Requests

Actor(s): Developer, Designer, Admin

- Function trigger: Developers, Designers, or Admins log into the Game Hub Platform and navigate to the Withdraw or Withdrawal Requests section from the main navigation menu to view submitted withdrawal requests.
- Function description: This feature enables Developers and Designers to track the status of their own withdrawal requests, while Admins can view and manage all user-submitted requests. The interface organizes requests by status (Pending, Completed, Rejected) and provides detailed request information to ensure clarity and transparency in the withdrawal process.
- Function details: This function displays a categorized list of withdrawal requests with filtering, searching, and detailed viewing options. Developers and Designers see only their own requests; Admins can access all. Each request includes essential metadata like amount, account details, and status. Admins can further export or process requests from this interface.
    - Business Rules:
        - Developers and Designers can only access their own withdrawal history.
        - Admins have full visibility and can search, filter, and process all requests.
        - Status is clearly marked as Pending, Approved, or Rejected.
        - Requests are read-only and cannot be edited once submitted.
        - Only Admins are allowed to take processing actions.
    - Abnormal cases:
        - No withdrawal requests exist: "No withdrawal requests found."
        - Request list fails to load: "Unable to load withdrawal requests. Please try again later."
        - Invalid search or filter input: Inline validation prompts specific to the field.
    - Success case:
        - Users see an up-to-date list of their withdrawal requests:
        - Developers/Designers: View request status and details under the Request History tab.
        - Admins: Filter, search, and export requests, or open a detailed view to take action.
        - Statuses update automatically in real time.

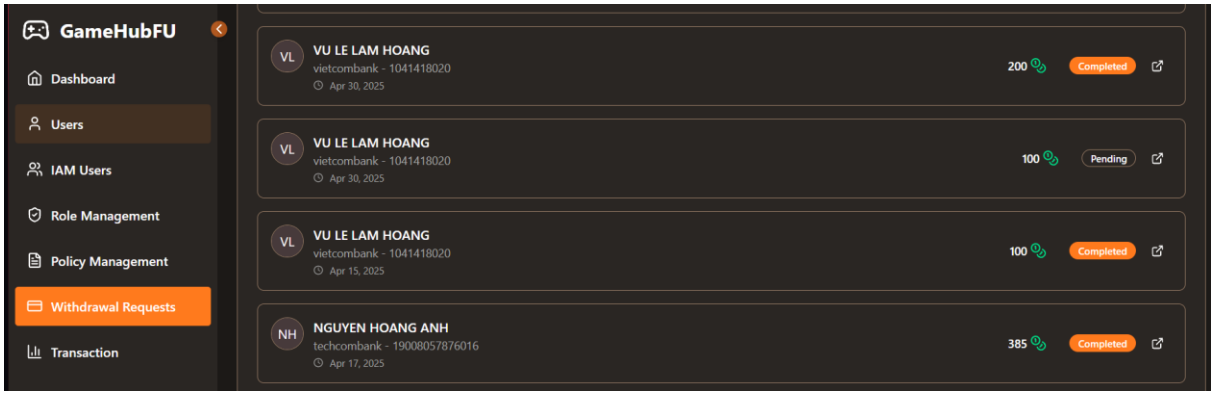

Figure 95 - Admin View The List Withrawal Requests History

3.16.4 View Withdrawal Request In Details

Actor(s): Admin, Developer, Designer

- Function trigger:
  - Developers, Designers, or Admins log into the Game Hub Platform, navigate to the Withdraw section (for Developers/Designers) or Withdrawal Requests section (for Admins), and click the View Details button/icon on a specific withdrawal request.
- Function description:
  - This feature provides a detailed view of withdrawal requests. Developers and Designers can review the full details of their own submissions. Admins can view any request and perform approval or rejection actions. The interface presents complete request data and processing options based on user roles.
- Function details:
  - This function renders a detailed view of a selected withdrawal request. Developers and Designers see a read-only breakdown of their submitted data. Admins have additional actions to approve or reject the request. The view includes status, request metadata, and a processing timeline for transparency and auditability.
- Business Rules:
  - Developers and Designers can only view their own withdrawal request details.
  - Admins can view and act on any withdrawal request.
  - Requests in "Approved" or "Rejected" state are locked from further edits or actions.
  - Admin approval must include a valid payment receipt.
  - Admin rejection must include a rejection reason.
  - All actions trigger confirmation dialogs to prevent accidental processing.
- Abnormal cases:
  - Request not found or unauthorized access →
    "Withdrawal request not found or access denied."
  - Error during approval/rejection →
    "An error occurred while processing the request. Please try again."
  - Missing rejection reason (Admin tries to reject without input) →
    "Please enter a rejection reason to proceed."
- Success case:
  - User successfully views full request details, including status and timeline.
  - Admin approves or rejects a request →
    Request status updates in real-time.
    Action is logged to the request timeline.
    The requester receives a system notification.
- Validation:

- User must be authenticated.
- Developers and Designers are restricted to viewing only their own requests.
- Admins must have processing permissions to act on requests.
- Rejecting a request requires a non-empty reason.
- Confirmation modals must be accepted before finalizing actions.

- Behavior notes:
  - Requests cannot be edited after submission.
  - Once approved or rejected, the request becomes read-only.
  - All Admin actions (approve/reject) are tracked and shown in the timeline.
  - Admins can continue to process other requests without needing to refresh the list.

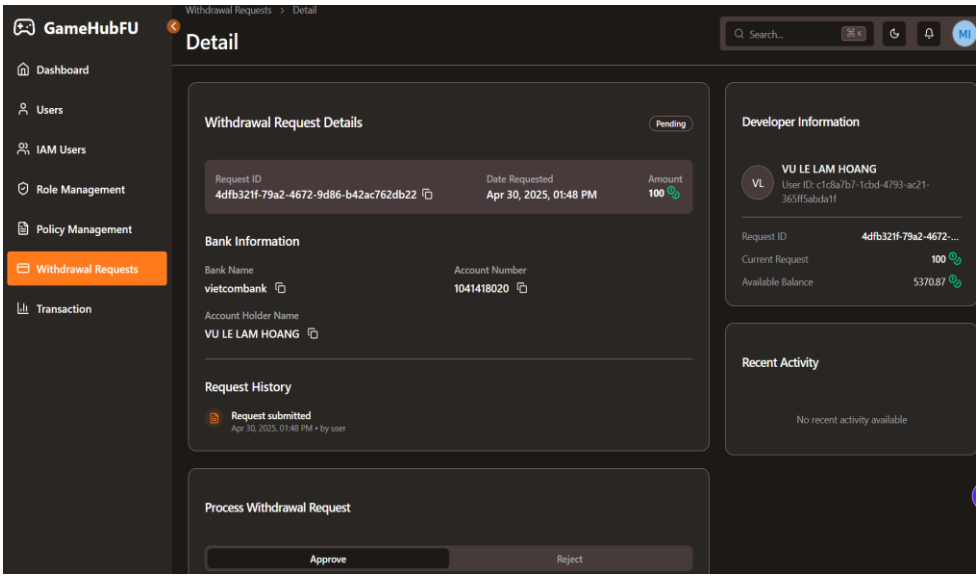

Figure 96 - Admin View The Withrawal Request In Details

3.16.5 Export Withdrawal Request Into File

Actor(s): Admin

- Function trigger: Admins log into the Game Hub Platform, navigate to the Diamond Transactions section from the main navigation menu.
- Function description: This feature provides Admins with access to a comprehensive list of all transactions related to the purchase of diamond packages. The transaction list helps Admins monitor platform revenue, user purchases, and potential discounts applied.
- Function details: This function returns a list of all transactions of purchased diamond packages that the Admin has access to, displayed in the form of a table, which includes:
  - Numerical order
  - Created time
  - Email
  - Package type
  - Discount
  - Price
  - Business Rules:
    - Only Admins can access this section.
    - All transaction records are read-only and cannot be modified.
    - Discounts, if applied, are displayed per transaction.
  - Abnormal cases:

- No transaction data found : "No diamond transactions available."
- Failed to load data : "Unable to load transactions. Please try again."

- Success case:
  - Transactions are fetched successfully and displayed in a paginated table.
  - Admin can scroll, filter, and sort the list as needed.

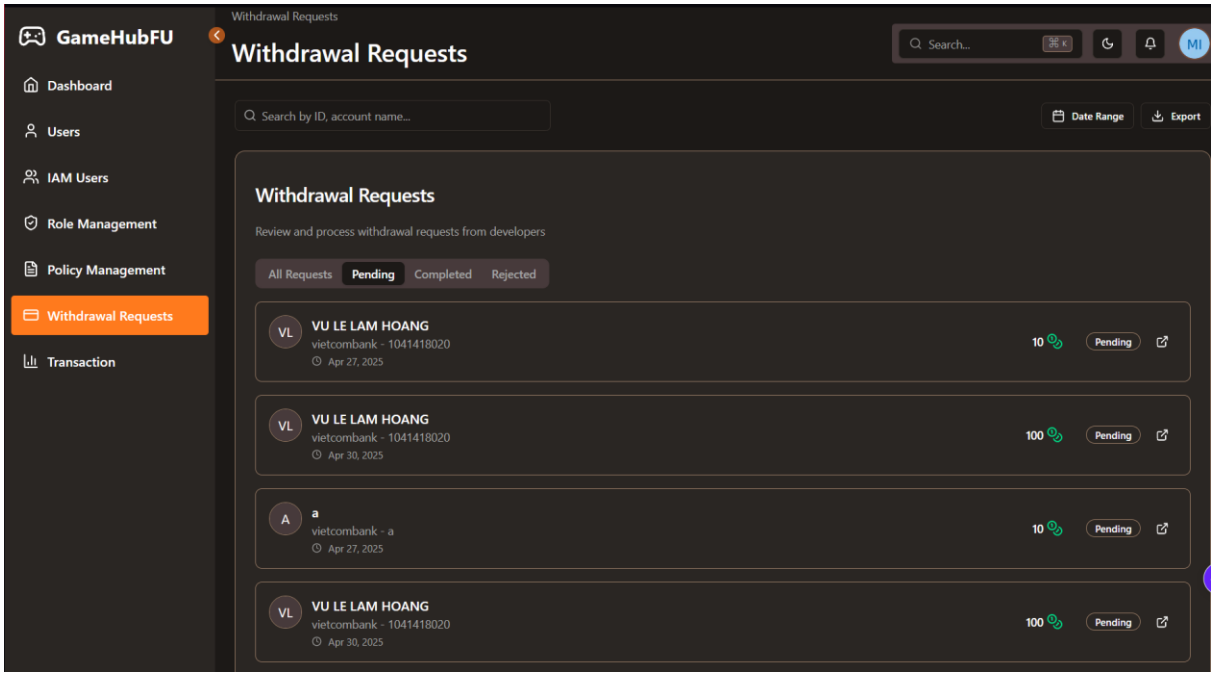

Figure 97 - Admin View The List Of Pending Withrawal Requests

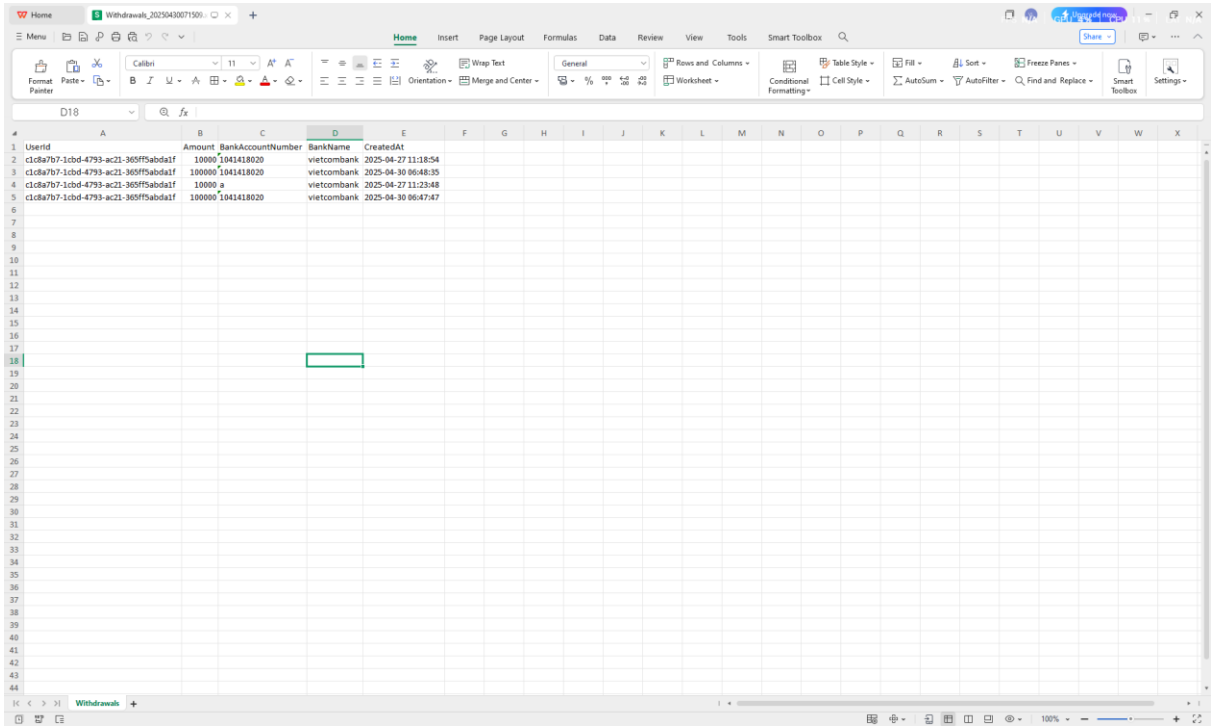

Figure 98 - Admin Exported Pending Withrawal Requests

3.17 News Management

3.17.1 Create News

Actor(s): Moderator

- Function trigger: Moderator clicks on the "Create News" button to open the **Create News** form.
- Function description: The Create News feature allows the Moderator to create, edit, and publish news articles within the platform. This functionality is located in the News section, where the Moderator can input the title, content, category, and associated image for the news article. The article can either be published immediately or saved as a draft for future editing. The Create News page provides an easy-to-use editor for writing the content and uploading a thumbnail image.
- Function details: This function opens a new page or modal to enter the title, content, and other details of the news article, and fill followings fields: Title, Content, Category, Thumbnail Image.

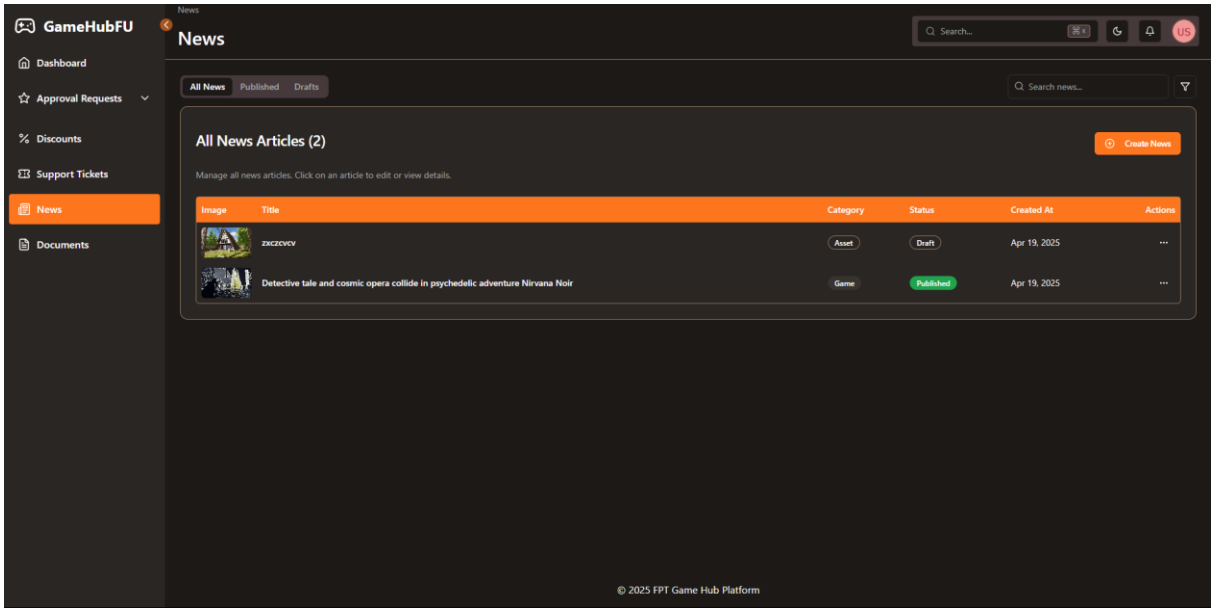

Figure 99 - News Entrance

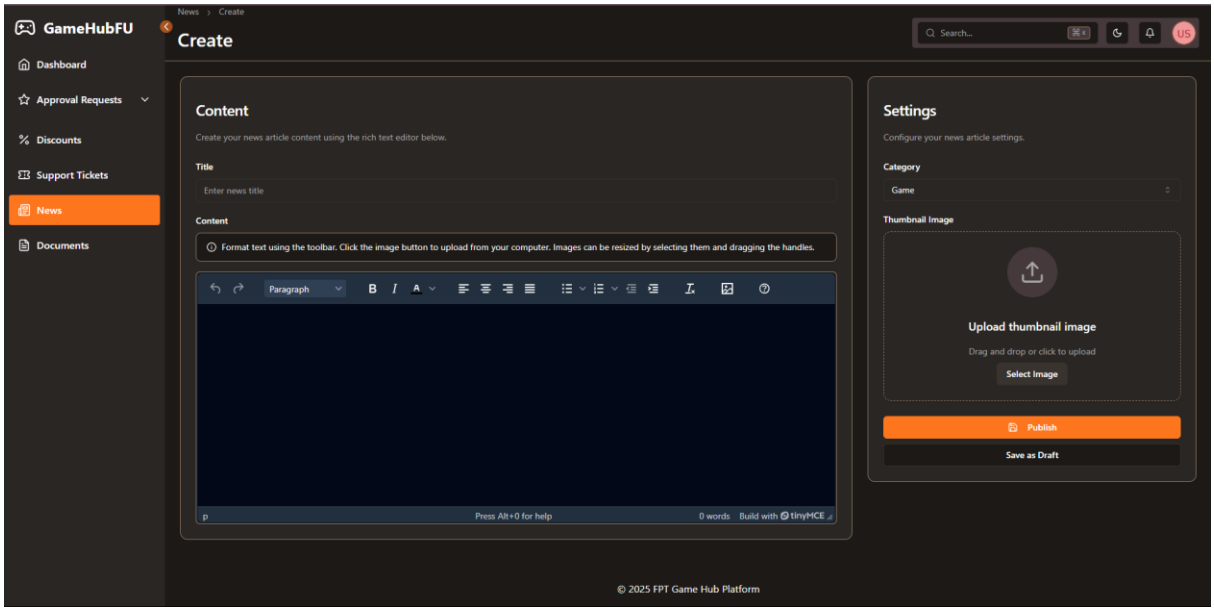

Figure 100 - Write New News

3.17.2 Browse News

- Function trigger: The user clicks on a news article headline or the Read More button.
- Function description: The Browse News function allows users to view news articles related to games and assets.
- Function details: The function navigates the user to the detailed view of the news article.

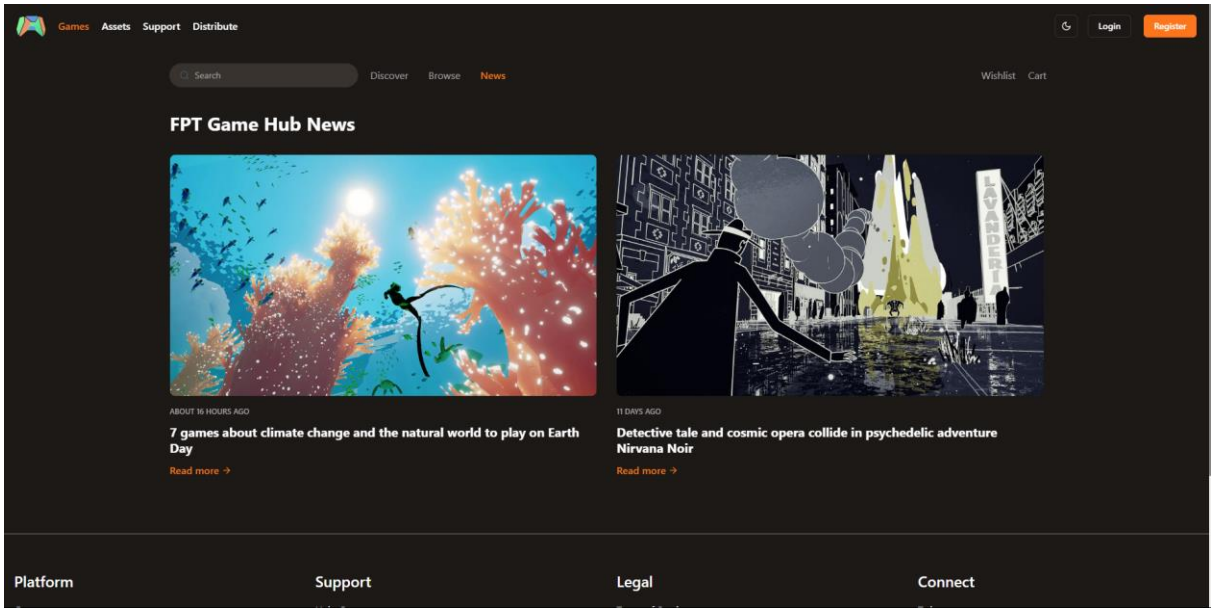

Figure 101 - View Publised News

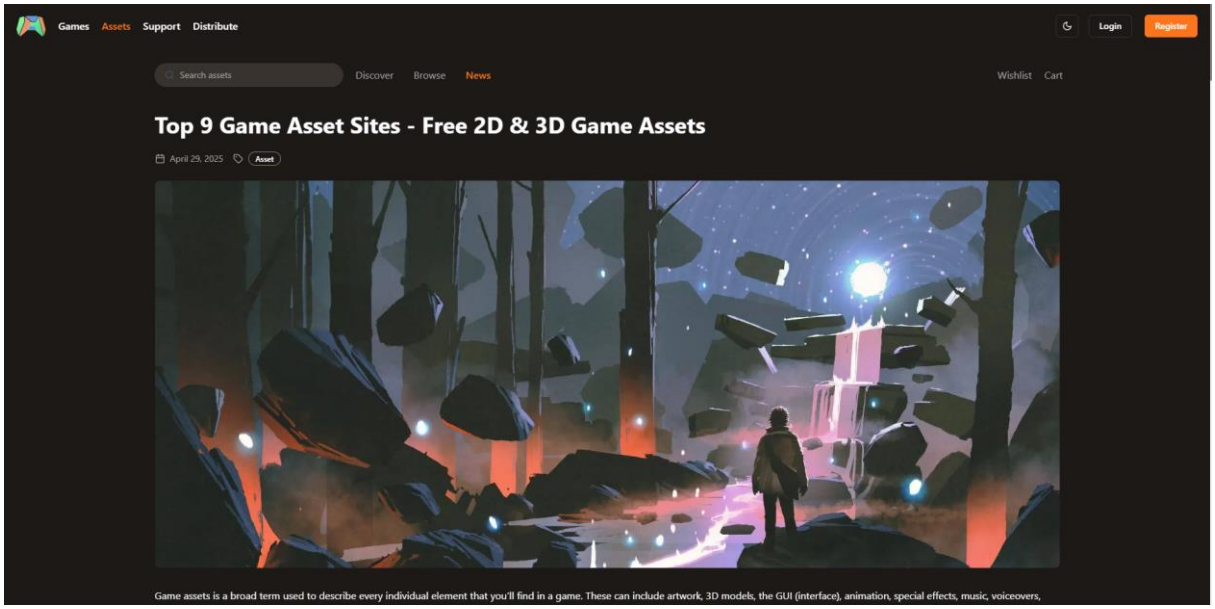

Figure 102 - Read News

3.18 Support Ticket Management

3.18.1 Send Support Ticket

Actor(s): Developer, Designer, Customer

- Function trigger: Customer, Developer, or Designer logs into the Game Hub Platform, navigates to the Support section, and submits a new support ticket via the support form.
- Function description: This feature allows platform users to report issues, request help, or provide feedback by submitting a support ticket. The form captures essential information such as issue type, description, and optional attachments. Once submitted, the ticket is routed to the support team for review and response.
- Function details: This function allows Customers, Developers, or Designers to submit a support request by providing a subject, selecting a category, describing the issue in detail, and optionally uploading relevant files. Upon submission, the ticket is logged into the support system and marked for review by the support team.
    - Business Rules:
        - Subject, Category, and Description fields are mandatory.
        - Users must be logged in with a valid Customer, Developer, or Designer account.
        - Attachments are optional but recommended for technical issues.
        - All submitted tickets are routed based on category priority.
    - Abnormal cases:
        - Missing required fields: "Please complete all required fields before submitting your request."
        - File too large or unsupported format: "Each file must be under 10MB and in a supported format."
        - Submission failure: "An error occurred while submitting your ticket. Please try again later."
    - Success case:
        - System validates inputs and confirms the submission:
        - "Your support ticket has been submitted successfully."
        - Users remain on the support page and may monitor status through platform notifications or email.

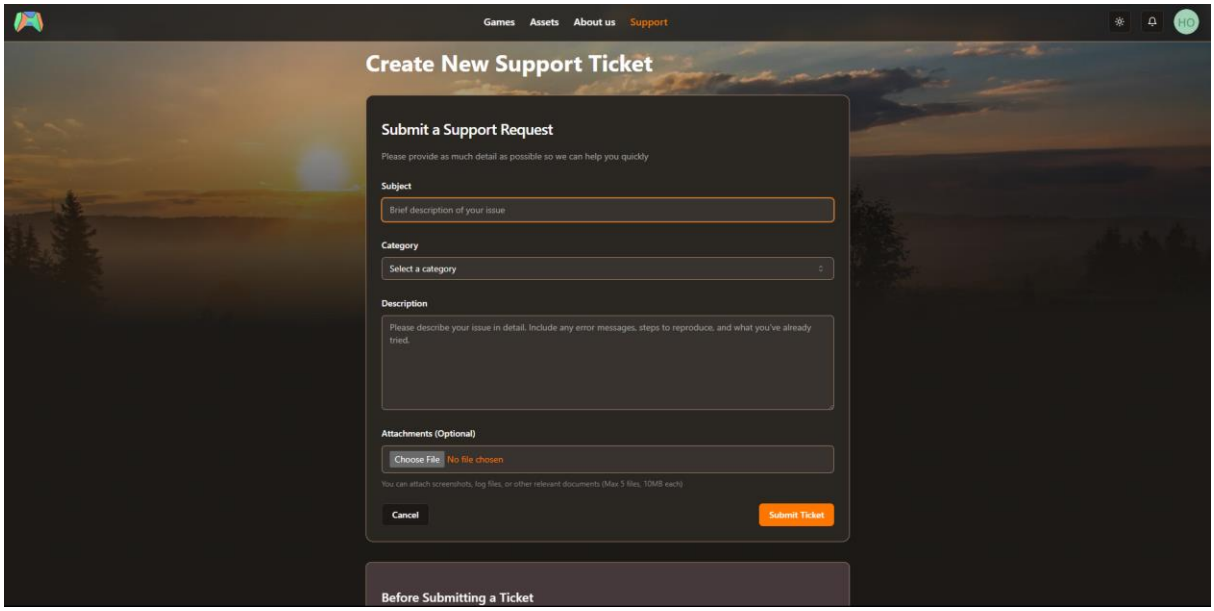

Figure 103 - Create New Support Ticket

3.18.2 Resolve Support Ticket

Actor(s): Moderator

- Function trigger: Moderator logs into the Game Hub Platform, navigates to the Support Tickets section, selects an open or in-progress ticket, and clicks the "Resolve Ticket" button after providing a response.
- Function description: This feature enables Moderators to review support tickets submitted by users, write or select a response, and resolve the issue. The system updates the ticket status and notifies the user, ensuring a structured and trackable support process.
- Function details: This function allows Moderators to resolve a support ticket by entering a required response—either through a predefined template or a custom message. Once submitted, the system marks the ticket as resolved and removes it from the active queue, notifying the original user of the resolution.
    - Business Rules:
        - Only users with a Moderator role can access and resolve tickets.
        - A response is mandatory before resolving a ticket.
        - Moderators may use a predefined response template or write their own message.
        - Once resolved, the ticket becomes read-only and cannot be modified.
        - Resolved tickets remain accessible for auditing and reference.
    - Abnormal cases:
        - No response entered → "Please enter a response before resolving the ticket."
        - System error during update → "An error occurred while resolving the ticket. Please try again."
    - Success case:
        - System accepts the moderator's input and updates the ticket status:
        - "Ticket resolved successfully."
        - The user is notified via platform notification or email.
        - The ticket is removed from the “In Progress” list.

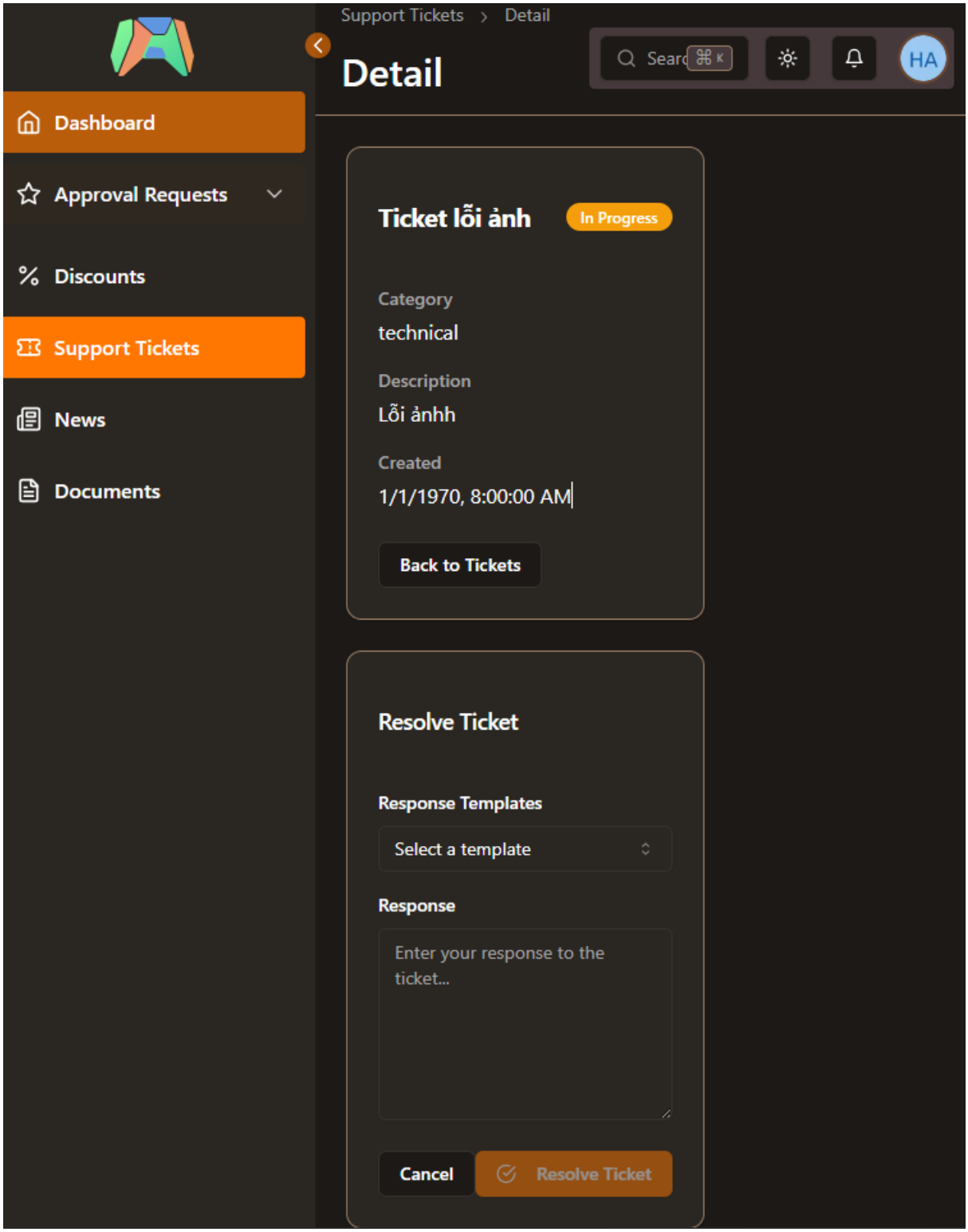

Figure 104 - Resolve Support Ticket

3.18.3 Track Support Tickets History

Actor(s): Developer, Designer, Customer, Moderator

- Function trigger: Users log into the Game Hub Platform, navigate to the Support Tickets section from the sidebar (for Moderators) or the top menu (for Developers, Designers, and Customers) to view their support ticket list grouped by status tabs.

- Function description: This feature enables all users to track the status of support tickets. Developers, Designers, and Customers can view their own submitted tickets and see Moderator responses, while Moderators have access to all incoming tickets for review and management. The interface includes filtering, sorting, and tabbed navigation to streamline ticket tracking.
- Function details: This function allows users to view support tickets categorized by status: All, In Progress, and Resolved. Each ticket displays essential information such as title, ID, creation date, and current status. Users can open ticket details to view the full message thread. Moderators have additional capabilities to search, filter, and manage all tickets.
    - Business Rules:
        - Users can only view tickets they submitted.
        - Moderators can view all tickets across the platform.
        - Tickets are automatically sorted by newest creation date first.
        - Status tabs dynamically reflect the ticket's current state.
        - Resolved tickets are read-only for all users.
        - Moderator responses are time stamped and shown chronologically.
    - Abnormal cases:
        - Ticket list fails to load: "Unable to retrieve support tickets. Please try again later."
    - No tickets match filters/search: "No tickets found matching your criteria."
    - Success case:
        - System loads and displays tickets with metadata and proper status grouping.
        - Users can open ticket details and view any responses or updates.
        - Tickets move automatically between tabs (e.g., from In Progress to Resolved) based on Moderator actions.

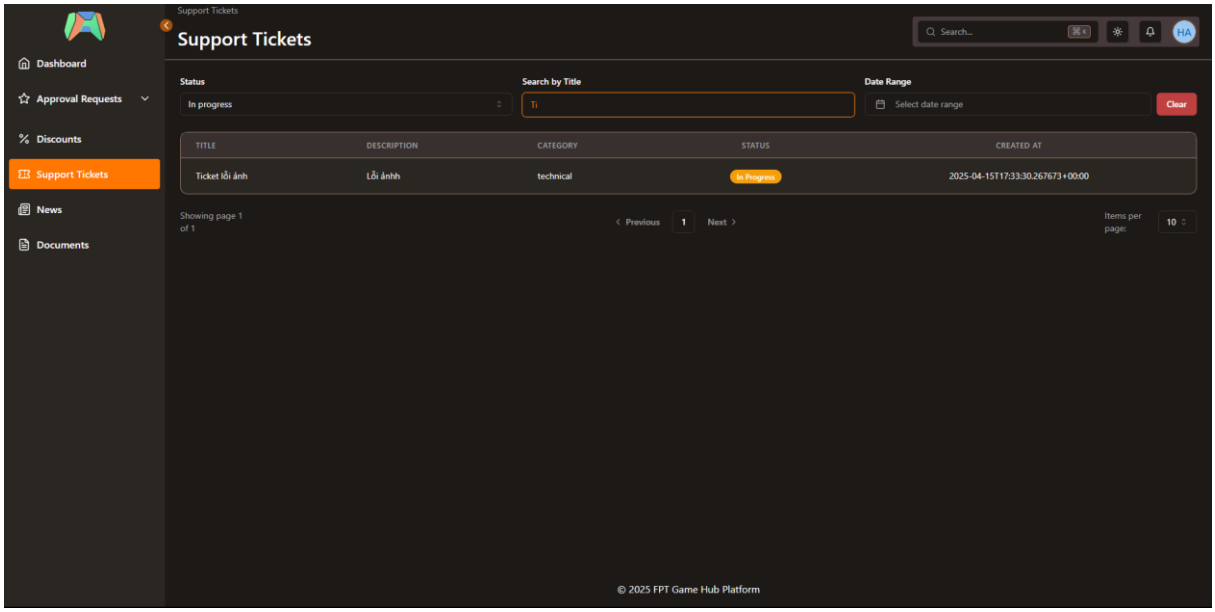

Figure 105 - Track Submitted Support Tickets List

4. Non-Functional Requirements

4.1 External Interfaces

4.1.1 User Interfaces

- UI-1: The languages used in the application is English
- UI-2: UI can be responsive on multiple screens, allowing the main content to display without scrolling horizontal appearance.

4.1.2 Communications Interfaces

- CI-1: HTTP ver1.1 Protocol is used for communication between web browsers, and servers.

4.2 Quality Attributes

- The platform must have a user-friendly interface with intuitive navigation and clear instructions, ensuring users can easily understand and use the system without the need for extensive training or expertise technique.

4.2.1 Portability

- Availability: The platform should have high availability. Downtime should be minimized, and mechanisms such as redundant servers should be in place to maintain uninterrupted service.

4.2.2 Usability

- Resource Utilization: The platform should optimize resource utilization, such as CPU, memory, and disk space, to ensure efficient performance. It should be designed to utilize system resources effectively, minimizing bottlenecks and maximizing overall system performance.
- Network Efficiency: The platform should be designed to minimize network latency and reduce the amount of data transmitted over the network. Efficient data transfer protocols and techniques, such as compression and caching, should be employed to optimize network performance.
- Database Performance: The platform's database should be optimized for performance, including efficient query execution, indexing, and data retrieval. Database operations should be fast and scalable to handle increasing data volumes without impacting the platform's responsiveness.
- Data Integrity: The platform should ensure data integrity and consistency, guaranteeing that user data is accurate and reliable. Data validation, proper error handling, and backup mechanisms should be implemented to prevent data corruption or loss.

4.2.3 Reusability

- The system shall be designed in a modular and component-based architecture to ensure that at least 80% of business logic components can be reused across multiple projects or features with minimal modification (no more than 20% code change).

4.2.4 Correctness

- The system shall produce accurate and expected results under all defined conditions, ensuring outputs strictly conform to the specified functional requirements, data formats, and business rules.

4.2.5 Maintainability

- Divide components based on the frameworks used for easy maintenance, upgrading, and debugging.

4.2.6 Reliability

- The system shall consistently perform its intended functions without failure under predefined conditions for a specified period of time, ensuring dependable operation and minimal disruptions.

4.2.7 Security

- Authorization by role.
- Data Protection: The platform should ensure the confidentiality, integrity, and availability of user data. Implement appropriate measures to protect against unauthorized access, use, disclosure, or modification of data.

- User Authentication: Users should be required to authenticate themselves using secure and reliable methods such as username/password combinations, or integration with existing authentication systems.
- Access Control: Implement role-based access control (RBAC) to restrict user access to specific features and data based on their roles and responsibilities. Ensure that users can only access the information necessary for their job functions.
- Encryption: Employ strong encryption algorithms to safeguard sensitive data during transmission and storage. This includes encrypting communication channels, user credentials, and any other sensitive information.
- Secure Communication: Use secure communication protocols (e.g., HTTPS) to protect data exchanged between the platform and users. This includes communication during authentication, data submission, and any other sensitive interactions.

5. Requirement Appendix

5.1 Business Rules

ID	Rule Definition
BR-01	Only FPT University students are allowed to register to become Developer or Designer of the GameHub Platform.
BR-02	Student card is required when students request FPTU student confirmation.
BR-03	Only the Moderator role is allowed to confirm, approve, or reject any approval requests.
BR-04	Only one Admin account is allowed in the system.
BR-05	Moderator and Admin do not need to register an account, but use the provided account to log in to the system.
BR-06	Guest, Developer, and Designer must use their registered account to log in to the system.
BR-07	Each account in the system is associated with only one role.
BR-08	Only the Developer role is allowed to create game products and game packages.
BR-09	Only the Designer role is allowed to create asset products and asset packages.
BR-10	Only approved game and asset packages can be showcased on the platform store.

BR-11	The Developer must approve the GameHub Developer Policy.
BR-12	The Designer must approve the GameHub Designer Policy.
BR-13	Only the Moderator role is allowed to ban or unban any content that violates the content standards policy.
BR-14	The system currency is coin.
BR-15	One coin in the system equals 1,000 VND in real money.
BR-16	Only the Developer role is allowed to create game projects and obtain API keys to integrate GameHub APIs.
BR-17	Developers must use API keys when calling GameHub API features.
BR-18	Only the Customer role is allowed to purchase digital packages.
BR-19	Customers must use their coin balance to purchase any asset or game packages.
BR-20	Customers must use real money to purchase coins through PayOS.
BR-21	Customers cannot purchase the same digital package more than once.
BR-22	Customers cannot add the same digital package to a wishlist more than once.
BR-23	Customers cannot repurchase already purchased digital packages.
BR-24	Customers must have sufficient coin balance before checking out an order.
BR-25	Customers are allowed to check out only one order at a time.
BR-26	Customers must go to their library to download purchased game or asset packages.
BR-27	Customers must not share their purchased packages if the package is not free.

BR-28	Customers, Developers, and Designers can only view wallet transactions related to them.
BR-29	Customers, Developers, and Designers can only view wallet information that belongs to them.
BR-30	Customers are allowed to leave feedback on packages.
BR-31	Customers are allowed to delete feedback only if it was written by them.
BR-32	Customers are allowed to update feedback only if it was written by them.
BR-33	Customers cannot refund the purchased packages,
BR-34	The Platform and Developer or Designer share profits based on the revenue share ratio.
BR-35	The Platform and the Developer share revenue at a 12:88 ratio, based on the final package price after deducting any discounts from either party.
BR-36	The Platform and the Designer share revenue at a 12:88 ratio, based on the final package price after deducting any discounts from either party.
BR-37	Customers, Developers, and Designers can send support tickets.
BR-38	Only the Moderator role is allowed to resolve support tickets.
BR-39	Only the Moderator role is allowed to create platform discounts.
BR-40	The discount unit is in percentage.
BR-41	When the discount is expressed as a percentage, the final price is calculated as: original price × (1 - discount value / 100).
BR-42	Package discounts are activated at the start of the event and automatically deactivated upon its expiration.
BR-43	Only Developer and Designer roles are allowed to send withdrawal requests.
BR-44	Developers and Designers can only send a withdrawal request if there is no existing pending request.

BR-45	Only the Admin role is allowed to view withdrawal requests from Developers and Designers.
BR-46	Only the Admin role is allowed to approve or reject withdrawal requests.
BR-47	The Admin must provide an evidence image when approving a withdrawal request.
BR-48	The Admin is allowed to view platform transactions only.
BR-49	The Admin is allowed to create new system accounts only.
BR-50	The Admin is allowed to assign permissions to roles only.
BR-51	The Admin is allowed to create new permissions only.
BR-52	The Admin is allowed to update user roles only.
BR-53	The Admin is allowed to view system users only.
BR-54	The Admin is allowed to create new roles only.
BR-55	The Players of a Developer's game project can purchase game items using their GameHub Customer account.

Table 18 - Business Rules

5.2 Common Requirements

ID	Requirements	Description
CR_01	User account management	Users only view their owned account profile, update account profile.
CR_02	Seperate UI workspace	Admin, Developer, Designer, Customer, Moderator, each role must have seperate workspace, ui screen.
CR_03	Authentication	All Users are qruired to login to access the system.
CR_04	Notification	All Users can receive, view the notifications delivery to them

Table 19 - Common Requirements

5.3 Application Messages List

#	Message Code	Message Type	Context	Content
1	MSG01	In red text below input field	Username	Username must be at least 3 characters.
2	MSG02	In red text below input field	Email	Invalid email format.
3	MSG03	In red text below input field	Password	Password must be at least 8 characters.
4	MSG04	In red text below input field	Password	Password must contain at least one uppercase letter.
5	MSG05	In red text below input field	Password	Password must contain at least one number.
6	MSG06	In red text below input field	Password	Password must contain at least one special character.
7	MSG07	In red text below input field	Password	Passwords do not match.
8	MSG08	In red text below input field	Phone Number	Invalid phone number.
9	MSG09	In red text below input field	Amount	Minimum withdrawal amount is 100 coins.
10	MSG10	In red text below input field	Bank Name	Bank name is required.
11	MSG11	In red text below	Account Number	Account number is required.

		input field		
12	MSG12	In red text below input field	Account Name	Account holder name is required.
13	MSG13	In red text below input field	Title	Title is required.
14	MSG14	In red text below input field	Content	Content is required.
15	MSG15	In red text below input field	Image	Thumbnail image is required.
16	MSG16	In red text below input field	Email	Email doesn't match. Please enter the correct email.
17	MSG17	In red text below input field	Rejection	Please provide a reason for rejection.
18	MSG18	In red text below input field	Response	Response text cannot be empty.
19	MSG19	In red text at form bottom	Form	Please fill in all required fields.
20	MSG20	In red text below upload button	Upload	Please upload a payment receipt before approving.
21	MSG21	In red text below input field	Reason	Please provide a reason for rejecting this withdrawal request.
22	MSG22	Red toast notification	Login	Please check your credentials and try again.

23	MSG23	Red toast notification	Login	Failed to login.
24	MSG24	Red toast notification	Registration	An unexpected error occurred during registration.
25	MSG25	Red toast notification	Profile	Failed to load user data.
26	MSG26	Red toast notification	Profile	Failed to update profile.
27	MSG27	Red toast notification	Upload	Failed to upload profile picture.
28	MSG28	Red toast notification	Withdrawal	You do not have enough coins to complete the withdrawal.
29	MSG29	Red toast notification	Withdrawal	Your withdrawal request has failed.
30	MSG30	Red toast notification	Package	Failed to create package. Please try again.
31	MSG31	Red toast notification	Review	Failed to submit review request.
32	MSG32	Red toast notification	Review	Failed to retract review request.
33	MSG33	Red toast notification	Review	Failed to resubmit package.
34	MSG34	Red toast notification	Download	Failed to download the file. Please try again.
35	MSG35	Red toast notification	News	Failed to create news article. Please try again.
36	MSG36	Red toast	News	Failed to update news article.

		notification		Please try again.
37	MSG37	Red toast notification	Ticket	Failed to load ticket details.
38	MSG38	Red toast notification	Ticket	Failed to resolve ticket.
39	MSG39	Red toast notification	Payment	Payment failed. Please try again later.
40	MSG40	Green toast notification	Login	Login successful!
41	MSG41	Green toast notification	Registration	Registration successful!
42	MSG42	Green toast notification	Profile	Your profile has been updated successfully.
43	MSG43	Green toast notification	Profile	Profile picture updated successfully.
44	MSG44	Green toast notification	Withdrawal	Your withdrawal request has been created successfully.
45	MSG45	Green toast notification	Package	Your package has been created successfully.
46	MSG46	Green toast notification	Review	Review request submitted successfully.
47	MSG47	Green toast notification	Review	Review request retracted successfully.
48	MSG48	Green toast notification	Review	Package resubmitted successfully.
49	MSG49	Green toast notification	Download	Download started.
50	MSG50	Green toast notification	News	News article published successfully.
51	MSG51	Green toast notification	News	News article updated successfully.
52	MSG52	Green toast notification	Ticket	Ticket has been resolved successfully.
53	MSG53	Green toast notification	Approval	Approval successful!
54	MSG54	Green toast notification	Rejection	Rejection successful!
55	MSG55	Green toast	Upload	Uploading...

		notification		
56	MSG56	Green toast notification	Upload	Uploading Source...
57	MSG57	Green toast notification	Review	You're about to retract your review request. This will remove it from the review queue.
58	MSG58	Green toast notification	Review	You're about to resubmit for review. Make sure you've addressed any previous feedback.
59	MSG59	Green toast notification	Download	Preparing file for download...
60	MSG60	Green toast notification	Confirmation	Are you sure you want to approve this account request?
61	MSG61	Green toast notification	Confirmation	Are you sure you want to resolve this ticket? This action cannot be undone.
62	MSG62	Green toast notification	Payment	Processing payment...
63	MSG63	Green toast notification	Withdrawal	Processing withdrawal request...

Table 20 - Application Messages List

IV. Software Design Description

1. System Design

1.1 System Architecture

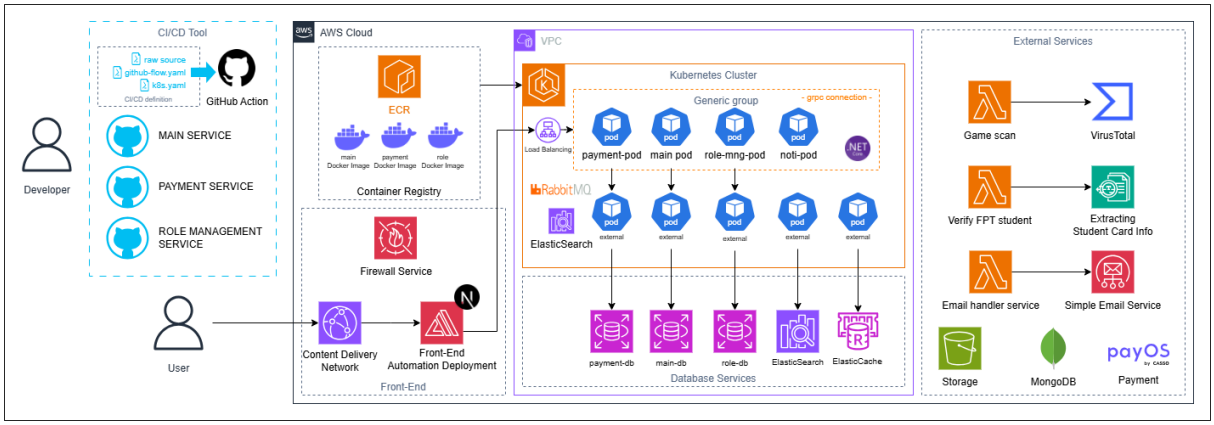

Figure 106 - System Architecture Diagram

1.2 Package Diagram

1.2.1 Frontend Package Diagram

1.2.1.1 Diagram

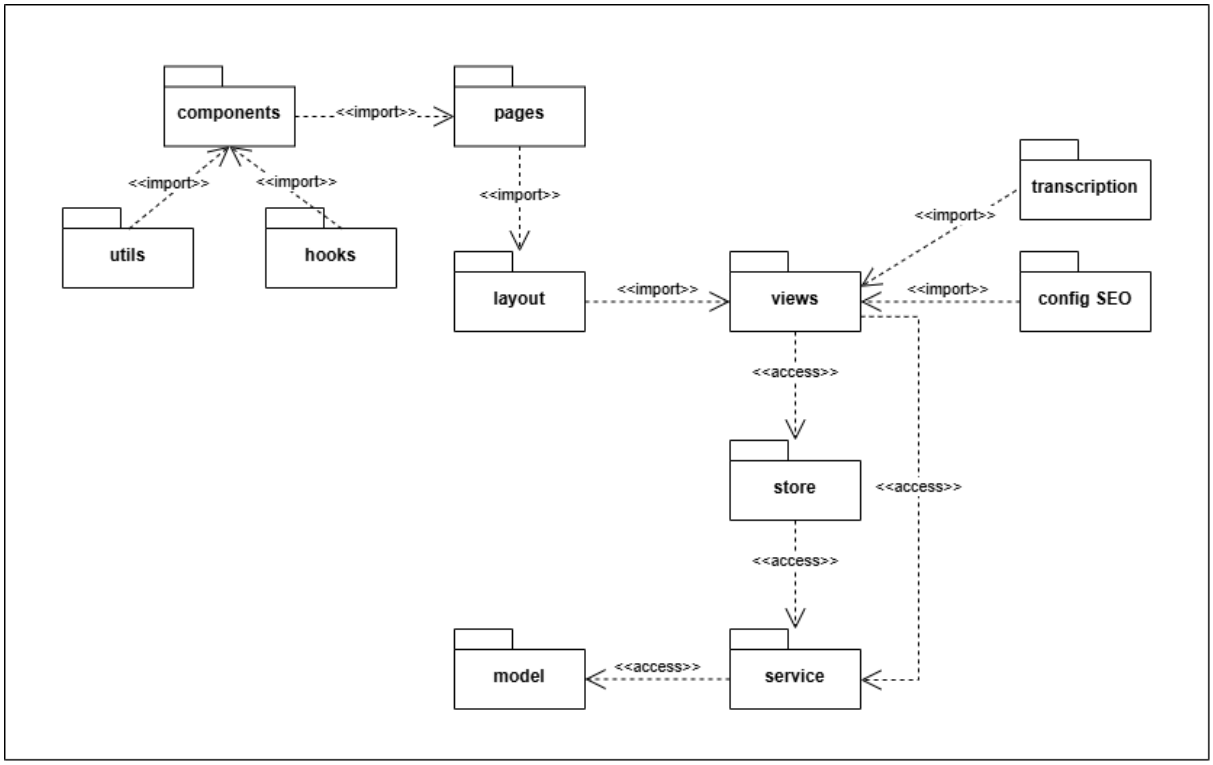

Figure 107 - Frontend Package Diagram

1.2.1.2 Package Description

This package diagram represents the structure and dependencies between major modules of the system.

components:

- Imports from utils and hooks.

utils and hooks:

- Provide utility functions and reusable hooks respectively, supporting components.

pages:

- Imports layout for page structure.

layout:

- Imports from views to render UI elements.

views:

- Imports from transcription and config SEO for features like transcription services and SEO configuration.
- Accesses store to manage application state.

store:

- Accesses service for business logic operations.

service:

- Access model to perform database or data-layer operations.
- Also interacts back to store to synchronize data if needed.

In summary, the package diagram can be separated to 3 main parts:

- *components, pages, and views* are the main UI layers.
- *store, service, and model* are the data management and business layers.
- *utils, hooks, transcription, and config SEO* provide supporting functions.

1.2.2 Backend Package Diagram

1.2.2.1 Game Hub Main Service Backend Package

1.2.2.1.1 Package Diagram

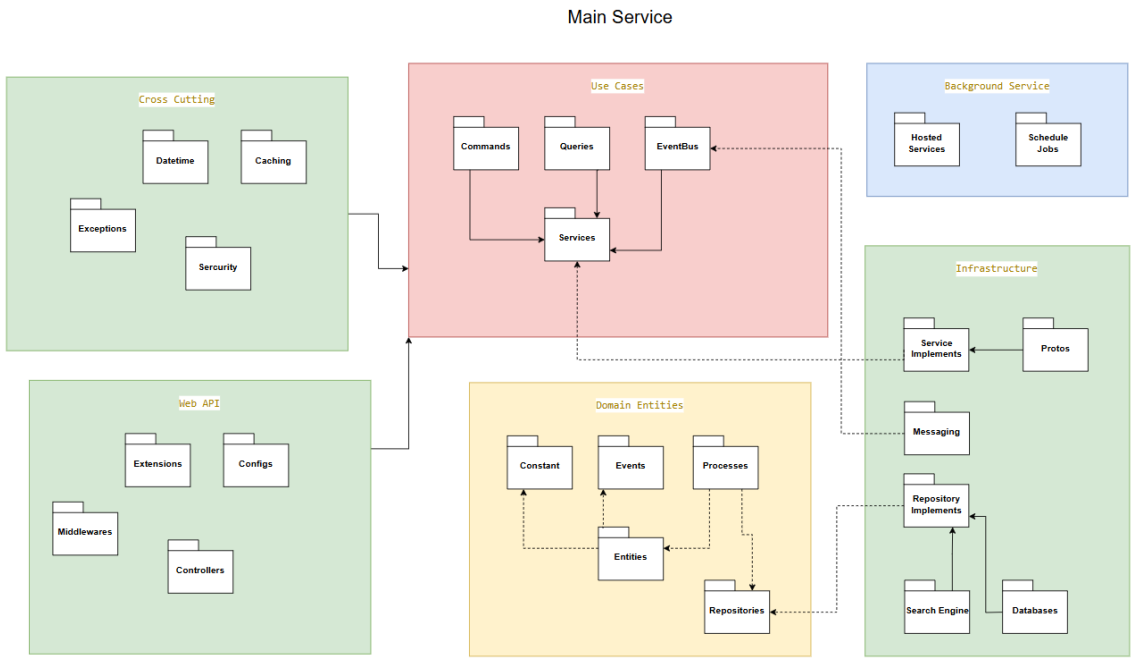

Figure 108 - Game Hub Main Service Backend Package Diagram

1.2.2.1.2 Package Layer Description

No.	Layer	Description
1	Use Cases	This is the heart of the business logic layer.
2	Domain Entities	Represents core business models and logic.
3	Infrastructure	Handles integrations with external systems and concrete implementations.

4	Background Service	Runs asynchronous tasks or recurring jobs.
5	Web API	Responsible for handling HTTP requests and exposing APIs.
6	Cross Cutting	Shared utilities and logic across layers.

Table 21 - GameHub Main Layer Description

1.2.2.1.3 Package Description

No.	Package	Description
1	Commands	Write operations (e.g., create, update, delete).
2	Queries	Read operations.
3	Services	Coordinate between commands/queries and domain logic.
4	EventBus	Publishes/subscribes domain events.
5	Entities	Domain objects (e.g., User, Order, etc.).
6	Repositories	Interfaces for persistence operations.
7	Processes	Business workflows or aggregates.
8	Events	Domain events.
9	Constant	Domain-level constants.

10	Service Implements	Concrete implementations of domain or application services.
11	Repository Implements	Implementation of Repositories using databases or search engines.
12	Messaging	Communication infrastructure (e.g., Kafka, RabbitMQ).
13	Search Engine	Integration with tools like Elasticsearch.
14	Databases	Connection and context setup.
15	Protos	gRPC or protocol buffer definitions.
16	Hosted Services	Long-running background processes.
17	Schedule Jobs	Scheduled (e.g., cron-based) tasks.
18	Controllers	Route HTTP requests to Use Cases.
19	Middlewares	Request/response interception (e.g., logging, auth).
20	Configs	Configuration binding.
21	Extensions	Utility extensions for the web layer.
22	Datetime	Time handling utilities (e.g., UTC, timezones).
23	Caching	In-memory/distributed caching helpers.

25	Security	JWT, encryption, access control.
26	Exceptions	Custom exceptions and handlers.

Table 22 - GameHub Main Package Description

1.2.2.2 Notification Service Backend Package

1.2.2.2.1 Package Diagram

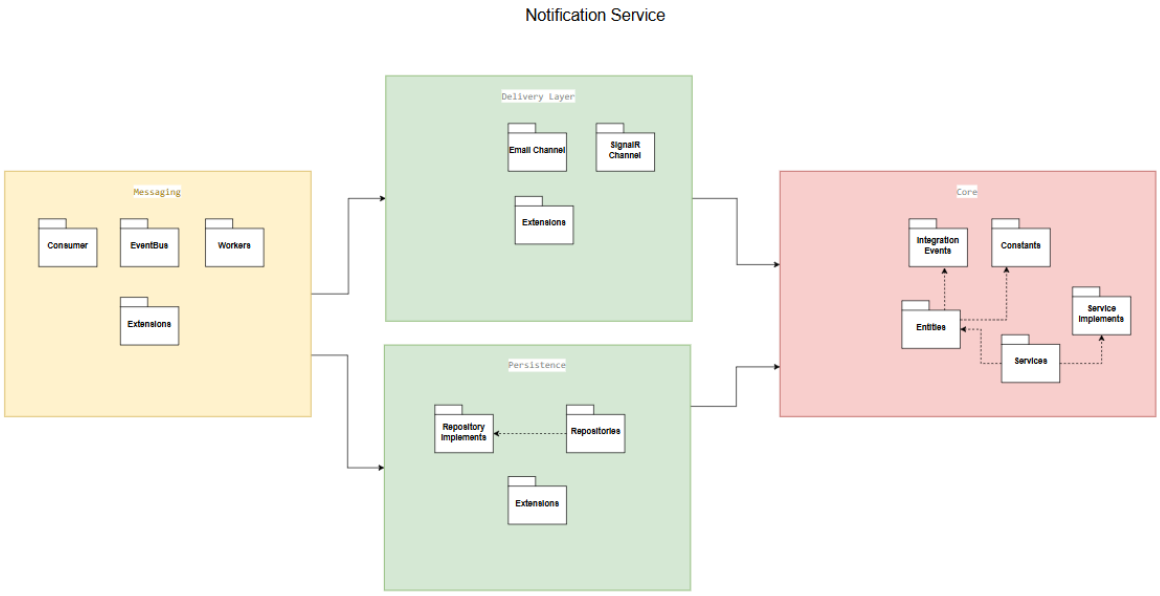

Figure 109 - Notification Service Backend Package Diagram

1.2.2.2.2 Package Layer Description

No.	Layer	Description
1	Messaging	Responsible for event-driven communication and background processing.
2	Delivery Layer	Handles actual message delivery to users via different channels.
3	Persistence	Manages storage and data access.
4	Core	Central business logic and domain layer.

Table 23 - Notification Layer Description

1.2.2.2.3 Package Description

No.	Package	Description
1	Consumer	Listens for incoming events from message brokers (e.g., RabbitMQ, Kafka).
2	EventBus	Facilitates event publishing and subscribing mechanisms.
3	Workers	Background services to handle long-running tasks (e.g., sending queued notifications).
4	Extensions	Helper functions or middleware for messaging configuration.
5	Email Channel	Sends notifications via email.
6	SignalR Channel	Pushes real-time updates using SignalR/WebSockets.
7	Extensions	Utility classes or custom DB context configurations.
8	Repository Implements	Concrete implementations of repository interfaces.
9	Repositories	Interfaces for data access (e.g., fetching or storing notification data).
10	Extensions	Utility classes for common operations (e.g., email formatting or connection handling).
11	Integration Events	Defines events triggered by the system (e.g., UserRegisteredEvent, MessageSentEvent).
12	Constants	Application-wide constant values.

13	Service Implements	Actual implementation of core services.
14	Services	Interfaces defining domain services (e.g., notification scheduler).
15	Entities	Domain models (e.g., Notification, UserPreference).

Table 24 - Notification Package Description

1.2.2.3 Payment Service Backend Package

1.2.2.3.1 Package Diagram

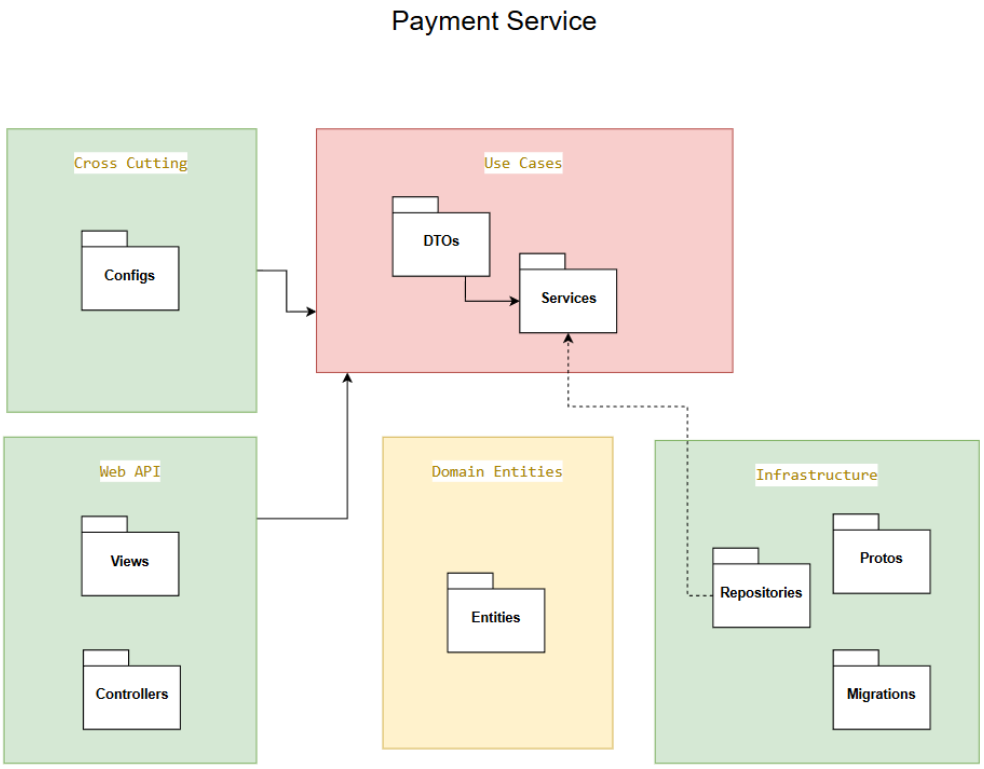

Figure 110 - Notification Service Backend Package Diagram

1.2.2.3.2 Package Layer Description

No.	Layer	Description
1	Web API Layer	The entry point for client interaction and HTTP requests.

2	Use Cases Layer	Encapsulates the application's core business logic and application services.
3	Domain Entities Layer	Contains the core domain models and business rules.
4	Infrastructure Layer	Provides technical implementation details such as database access and external services.

Table 25 - Payment Layer Description

1.2.2.3.3 Package Description

No.	Package	Description
1	Configs	Contains application-wide configurations (e.g., dependency injection, logging, environment variables).
2	Controllers	Contains the API endpoints that receive requests, validate input, and delegate business logic to the BLL layer.
3	Views	Represents view models or response shaping logic (if applicable).
4	DTOs	Defines data contracts used for transferring data between layers.
5	Services	Implements business workflows (e.g., processing payments, validating transactions).
6	Entities	Represents the core business objects (e.g., Payment, Transaction, UserAccount) and their behaviors.
7	Protos	May include protocol buffer definitions if gRPC is used, or shared message schemas.
8	Repositories	Encapsulates data access logic, typically implementing the Repository pattern.

Table 26 - Payment Package Description

1.2.2.4 Role-base Service Backend Package

1.2.2.4.1 Package Diagram

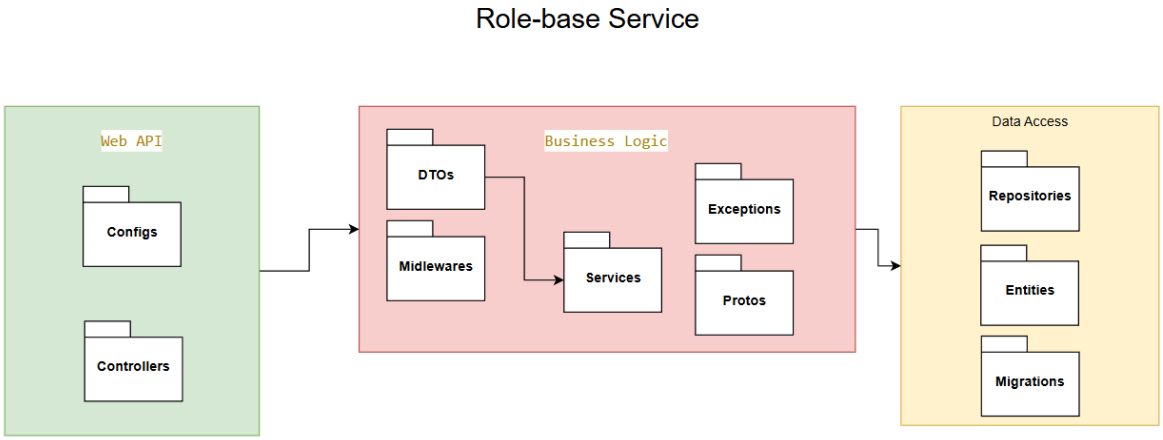

Figure 111 - Role-base Service Backend Package Diagram

1.2.2.4.2 Package Layer Description

No.	Layer	Description
1	Web API Layer	The entry point for client interaction and HTTP requests.
2	Business Logic Layer	The core of the system, encapsulating all business rules and operations.
3	Data Access Layer	Responsible for interaction with the database.

1.2.2.4.3 Package Description

No.	Package	Description
1	Configs	Handles configuration files such as appsettings.json, environment settings, and dependency injection setup.
2	Controllers	Contains the API endpoints that receive requests, validate input, and delegate business logic to the BLL layer.
3	DTOs	Defines data transfer objects used between layers, especially between API and services.

4	Middlewares	Custom middleware components for request/response processing, like exception handling or logging.
5	Services	Implements the application’s core logic, such as role validation, permission assignment, and user access control.
6	Exceptions	Centralized handling of application-specific exceptions.
7	Protos	May include protocol buffer definitions if gRPC is used, or shared message schemas.
8	Repositories	Encapsulates data access logic, typically implementing the Repository pattern.
9	Entities	Contains the domain models that map to database tables (often using EF Core or another ORM).
10	Migrations	Includes EF Core migration files that manage schema changes over time.

2. Database Design

2.1 Database Design

2.1.1 Physical SQL Database Design (MySQL)

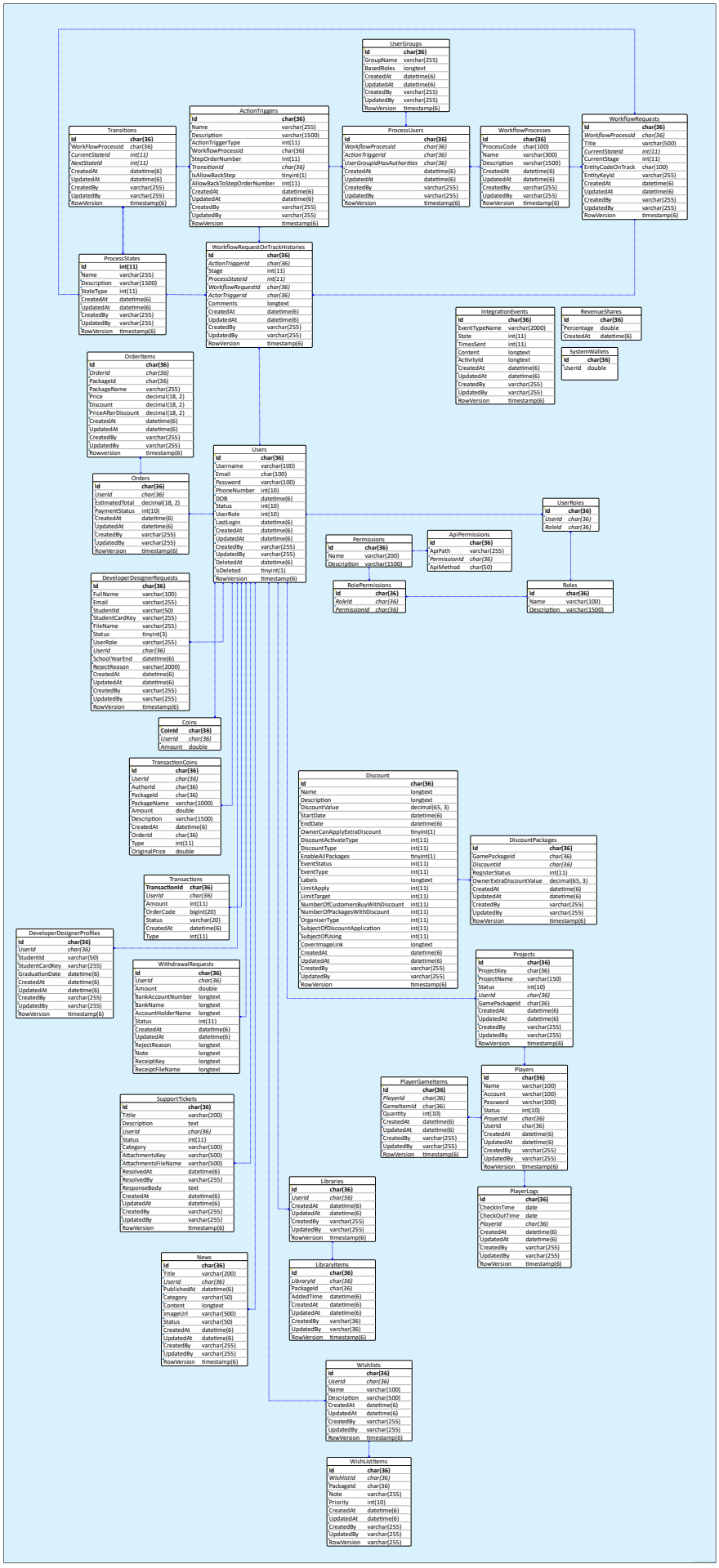
UserGroups
Id char(36)
GroupName varchar(255)
BasedRoles longtext
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
ActionTriggers
Id char(36)
Name varchar(255)
Description varchar(1500)
ActionTriggerType int(11)
WorkflowProcessId char(36)
StepOrderNumber int(11)
TransitionId char(36)
IsAllowBackStep tinyint(1)
AllowBackToStepOrderNumber int(11)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
ProcessUsers
Id char(36)
WorkflowProcessId char(36)
ActionTriggerId char(36)
UserGroupIdHasAuthorities char(36)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
WorkflowProcesses
Id char(36)
ProcessCode char(100)
Name varchar(300)
Description varchar(1500)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
WorkflowRequests
Id char(36)
WorkflowProcessId char(36)
Title varchar(500)
CurrentStateId int(11)
CurrentStage int(11)
EntityCodeOnTrack char(100)
EntityKeyId varchar(255)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
Transitions
Id char(36)
WorkFlowProcessId char(36)
CurrentStateId int(11)
NextStateId int(11)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
ProcessStates
Id int(11)
Name varchar(255)
Description varchar(1500)
StateType int(11)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
WorkflowRequestOnTrackHistories
Id char(36)
ActionTriggerId char(36)
Stage int(11)
ProcessStateId int(11)
WorkflowRequestId char(36)
ActorTriggerId char(36)
Comments longtext
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
IntegrationEvents
Id char(36)
EventTypeName varchar(2000)
State int(11)
TimesSent int(11)
Content longtext
ActivityId longtext
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
RevenueShares
Id char(36)
Percentage double
CreatedAt datetime(6)
SystemWallets
Id char(36)
UserId double
OrderItems
Id char(36)
OrderId char(36)
PackageId char(36)
PackageName varchar(255)
Price decimal(18, 2)
Discount decimal(18, 2)
PriceAfterDiscount decimal(18, 2)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
Rowversion timestamp(6)
Users
Id char(36)
Username varchar(100)
Email char(100)
Password varchar(100)
PhoneNumber int(10)
DOB datetime(6)
Status int(10)
UserRole int(10)
LastLogin datetime(6)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
DeletedAt datetime(6)
IsDeleted tinyint(1)
RowVersion timestamp(6)
Orders
Id char(36)
UserId char(36)
EstimatedTotal decimal(18, 2)
PaymentStatus int(10)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
UserRoles
Id char(36)
UserId char(36)
RoleId char(36)
Permissions
Id char(36)
Name varchar(200)
Description varchar(1500)
ApiPermissions
Id char(36)
ApiPath varchar(255)
PermissionId char(36)
ApiMethod char(50)
RolePermissions
Id char(36)
RoleId char(36)
PermissionId char(36)
Roles
Id char(36)
Name varchar(100)
Description varchar(1500)
DeveloperDesignerRequests
Id char(36)
FullName varchar(100)
Email varchar(255)
StudentId varchar(50)
StudentCardKey varchar(255)
FileName varchar(255)
Status tinyint(3)
UserRole varchar(255)
UserId char(36)
SchoolYearEnd datetime(6)
RejectReason varchar(2000)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
Coins
CoinId char(36)
UserId char(36)
Amount double
TransactionCoins
Id char(36)
UserId char(36)
AuthorId char(36)
PackageId char(36)
PackageName varchar(1000)
Amount double
Description varchar(1500)
CreatedAt datetime(6)
OrderId char(36)
Type int(11)
OriginalPrice double
Transactions
TransactionId char(36)
UserId char(36)
Amount int(11)
OrderCode bigint(20)
Status varchar(20)
CreatedAt datetime(6)
Type int(11)
Discount
Id char(36)
Name longtext
Description longtext
DiscountValue decimal(65, 3)
StartDate datetime(6)
EndDate datetime(6)
OwnerCanApplyExtraDiscount tinyint(1)
DiscountActivateType int(11)
DiscountType int(11)
EnableAllPackages tinyint(1)
EventStatus int(11)
EventType int(11)
Labels longtext
LimitApply int(11)
LimitTarget int(11)
NumberOfCustomersBuyWithDiscount int(11)
NumberOfPackagesWithDiscount int(11)
OrganiserType int(11)
SubjectOfDiscountApplication int(11)
SubjectOfUsing int(11)
CoverImageLink longtext
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
DiscountPackages
Id char(36)
GamePackageId char(36)
DiscountId char(36)
RegisterStatus int(11)
OwnerExtraDiscountValue decimal(65, 3)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
DeveloperDesignerProfiles
Id char(36)
UserId char(36)
StudentId varchar(50)
StudentCardKey varchar(255)
GraduationDate datetime(6)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
WithdrawalRequests
Id char(36)
UserId char(36)
Amount double
BankAccountNumber longtext
BankName longtext
AccountHolderName longtext
Status int(11)
CreatedAt datetime(6)
UpdatedAt datetime(6)
RejectReason longtext
Note longtext
ReceiptKey longtext
ReceiptFileName longtext
Projects
Id char(36)
ProjectKey char(36)
ProjectName varchar(150)
Status int(10)
UserId char(36)
GamePackageId char(36)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
SupportTickets
Id char(36)
Title varchar(200)
Description text
UserId char(36)
Status int(11)
Category varchar(100)
AttachmentsKey varchar(500)
AttachmentsFileName varchar(500)
ResolvedAt datetime(6)
ResolvedBy varchar(255)
ResponseBody text
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
PlayerGameItems
Id char(36)
PlayerId char(36)
GameItemId char(36)
Quantity int(10)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
Players
Id char(36)
Name varchar(100)
Account varchar(100)
Password varchar(100)
Status int(10)
ProjectId char(36)
UserId char(36)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
Libraries
Id char(36)
UserId char(36)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
PlayerLogs
Id char(36)
CheckInTime date
CheckOutTime date
PlayerId char(36)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
LibraryItems
Id char(36)
LibraryId char(36)
PackageId char(36)
AddedTime datetime(6)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(36)
UpdatedBy varchar(36)
RowVersion timestamp(6)
News
Id char(36)
Title varchar(200)
UserId char(36)
PublishedAt datetime(6)
Category varchar(50)
Content longtext
ImageUrl varchar(500)
Status varchar(50)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
Wishlists
Id char(36)
UserId char(36)
Name varchar(100)
Description varchar(500)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion timestamp(6)
WishListItems
Id char(36)
WishlistId char(36)
PackageId char(36)
Note varchar(255)
Priority int(10)
CreatedAt datetime(6)
UpdatedAt datetime(6)
CreatedBy varchar(255)
UpdatedBy varchar(255)
RowVersion varchar(255)

Figure 112 - Physical SQL Database

2.1.2 Physical NoSQL Database Design (Mongo)

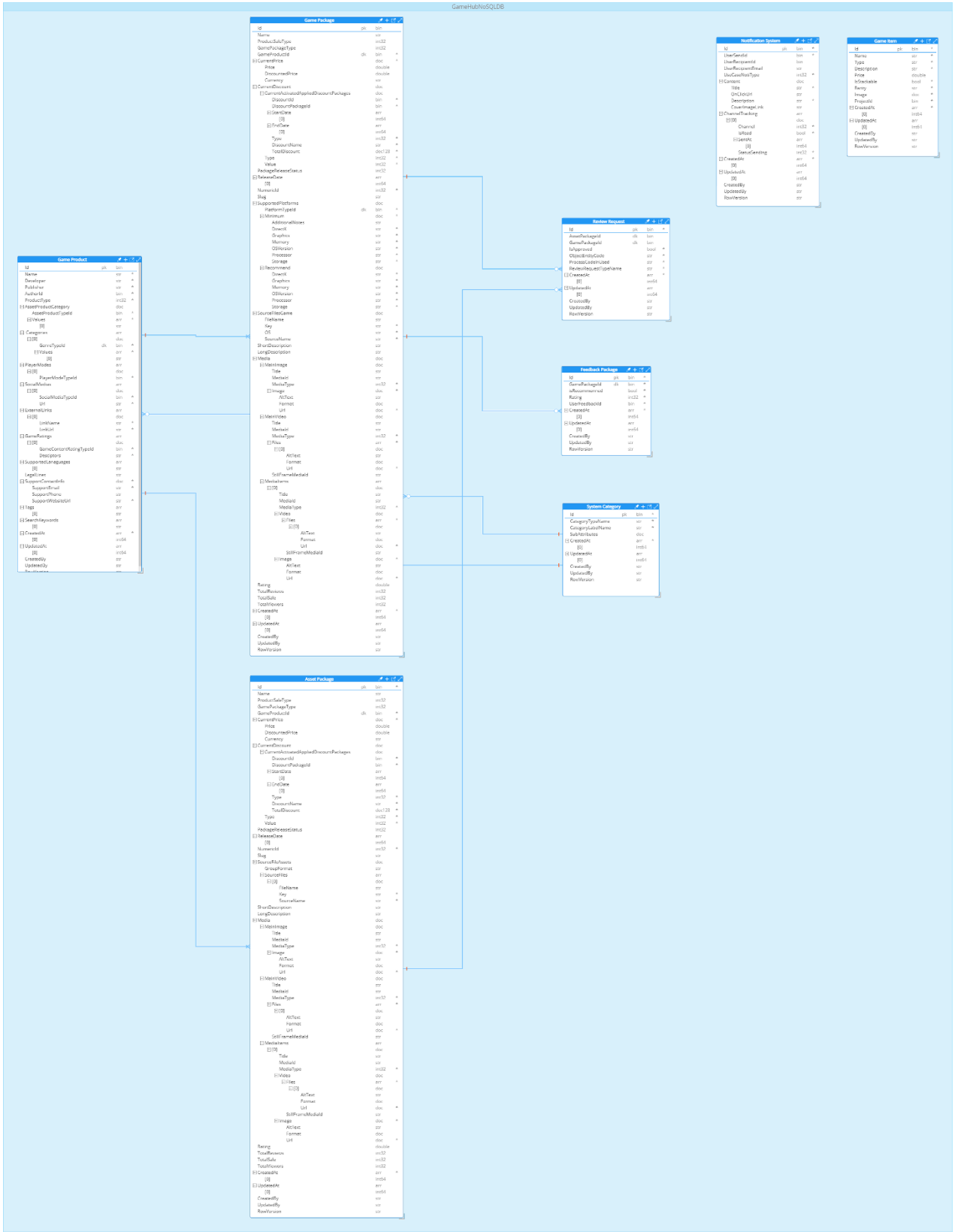

Figure 113 - Physical NoSQL Database

2.2 Table Description

No.	Table	Description	Store In Database Type
1	Users	Stores all user accounts registered on the platform.	SQL
2	DeveloperDesignerProfiles	Stores extended profiles for users registered as Developers or Designers.	SQL
3	DeveloperDesignerRequests	Tracks user-submitted requests to become Developers or Designers.	SQL
4	Discounts	Defines platform-wide discounts, including percentage, active duration, and conditions.	SQL
5	DiscountPackages	Links discounts to specific game products or packages.	SQL
6	IntegrationEvents	Stores domain integration events for asynchronous processing or external systems.	SQL
7	News	Contains news posts and announcements published by the platform.	SQL
8	Orders	Stores purchase orders made by users.	SQL
9	OrderItems	Contains individual items within a user’s order.	SQL
10	Wishlists	Represents collections of items users want to follow or purchase later.	SQL
11	WishlistItems	Lists the specific items saved within a user's wishlist.	SQL
12	Libraries	Links users to the game products and packages they have purchased or claimed.	SQL

13	LibraryItems	Stores individual game or asset items within a user's library.	SQL
14	Projects	Represents game integration projects managed by developers.	SQL
15	Players	Represents users' in-game identity and current status.	SQL
16	PlayerLogs	Records activity logs and in-game actions of players.	SQL
17	PlayerGameItems	Stores in-game items owned or used by players.	SQL
18	WorkflowProcesses	Defines reusable structured workflows for approvals or publishing.	SQL
19	WorkflowRequests	Tracks each workflow instance initiated by users.	SQL
20	WorkflowRequestOnTrackHistories	Stores the history of steps taken during the lifecycle of a workflow request.	SQL
21	Transitions	Defines allowed transitions between workflow states.	SQL
22	ActionTriggers	Represents actions that trigger state transitions in workflows.	SQL
23	ProcessUsers	Maps user groups to action triggers for specific workflow processes.	SQL
24	ProcessStates	Defines all possible states in a workflow process.	SQL
25	UserGroups	Stores custom-defined user groups for access control or communication.	SQL
26	SupportTickets	Records user-submitted support issues and related messages.	SQL

27	Coins	Stores the current coin balance of each user.	SQL
28	TransactionCoins	Logs transactions related to coins, including purchases and adjustments.	SQL
29	Transactions	Logs real-money financial transactions on the platform.	SQL
30	SystemWallets	Represents the platform's main wallet for handling financial operations.	SQL
31	WithdrawalRequests	Tracks withdrawal requests submitted by users for coins or earnings.	SQL
32	RevenueShares	Defines revenue-sharing agreements between users and the platform.	SQL
33	Roles	Lists available user roles, such as Admin, Developer, and Customer.	SQL
34	UserRoles	Maps users to their assigned roles.	SQL
35	RolePermissions	Links roles to the permissions they are allowed to perform.	SQL
36	Permissions	Defines the full list of permission codes available across the platform.	SQL
37	ApiPermissions	Manages permissions and access rules for API clients.	SQL
38	Game Package	Represents purchasable game packages including pricing and metadata.	NoSQL
39	Game Product	Represents core game entries with metadata, versioning, and categories.	NoSQL
40	Notification System	Stores all system-generated notifications sent to users.	NoSQL

41	Review Request	Tracks review or moderation requests, usually for submitted content or games.	NoSQL
42	System Category	Defines standardized tags and categories for organizing products and assets.	NoSQL
43	Game Item	Represents in-game virtual items such as cosmetics, power-ups, or consumables.	NoSQL
44	Asset Package	Represents purchasable asset packages such as models, sounds, or textures.	NoSQL
45	Feedback Package	Stores user-submitted feedback or reviews about packages.	NoSQL

Table 27 - Table Description

2.3 Attribute Data Dictionary

2.3.1 Table Users

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Username	varchar	100	FALSE	FALSE	FALSE
Email	char	100	FALSE	FALSE	FALSE
Password	varchar	100	FALSE	FALSE	FALSE
PhoneNumber	int	10	FALSE	TRUE	FALSE
DOB	datetime	6	FALSE	TRUE	FALSE
Status	int	10	FALSE	FALSE	FALSE
UserRole	int	10	FALSE	FALSE	FALSE
LastLogin	datetime	6	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
DeletedAt	datetime	6	FALSE	TRUE	FALSE

isDeleted	tinyint	1	FALSE	FALSE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 28 - Table Users

2.3.2 Table DeveloperDesignerProfiles

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
StudentId	varchar	50	FALSE	FALSE	FALSE
StudentCardKey	varchar	255	FALSE	FALSE	FALSE
GraduationDate	datetime	6	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 29 - Table DeveloperDesignerProfiles

2.3.3 Table DeveloperDesignerRequests

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
FullName	varchar	100	FALSE	FALSE	FALSE
Email	varchar	255	FALSE	FALSE	FALSE
StudentId	varchar	50	FALSE	FALSE	FALSE
StudentCardKey	varchar	255	FALSE	FALSE	FALSE
FileName	varchar	255	FALSE	FALSE	FALSE
Status	tinyint	3	FALSE	FALSE	FALSE
UserRole	varchar	255	FALSE	FALSE	FALSE
UserId	char	36	FALSE	FALSE	FALSE
SchoolYearEnd	datetime	6	FALSE	FALSE	FALSE
RejectReason	varchar	2000	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	TRUE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE

CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 30 - Table DeveloperDesignerRequests

2.3.4 Table Discounts

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Name	longtext	0	FALSE	FALSE	FALSE
Description	longtext	0	FALSE	FALSE	FALSE
DiscountValue	decimal	65	FALSE	FALSE	FALSE
StartDate	datetime	6	FALSE	FALSE	FALSE
EndDate	datetime	6	FALSE	TRUE	FALSE
OwnerCanApplyExtraDiscount	tinyint	1	FALSE	FALSE	FALSE
DiscountActivateType	int	11	FALSE	FALSE	FALSE
DiscountType	int	11	FALSE	FALSE	FALSE
EnableAllPackages	tinyint	1	FALSE	FALSE	FALSE
EventStatus	int	11	FALSE	FALSE	FALSE
EventType	int	11	FALSE	FALSE	FALSE
Labels	longtext	0	FALSE	TRUE	FALSE
LimitApply	int	11	FALSE	TRUE	FALSE
LimitTarget	int	11	FALSE	TRUE	FALSE
NumberOfCustomersBuyWithDiscount	int	11	FALSE	FALSE	FALSE
NumberOfPackagesWithDiscount	int	11	FALSE	FALSE	FALSE
OrganiserType	int	11	FALSE	FALSE	FALSE
SubjectOfDiscountApplication	int	11	FALSE	FALSE	FALSE
SubjectOfUsing	int	11	FALSE	TRUE	FALSE
CoverImageLink	longtext	0	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE

UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 31 - Table Discounts

2.3.5 Table DiscountPackages

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
GamePackageId	char	36	FALSE	FALSE	FALSE
DiscountId	char	36	FALSE	FALSE	FALSE
RegisterStatus	int	11	FALSE	TRUE	FALSE
OwnerExtraDiscountValue	decimal	65	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 32 - Table DiscountPackages

2.3.6 Table IntegrationEvents

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
EventTypeName	varchar	2000	FALSE	FALSE	FALSE
State	int	11	FALSE	FALSE	FALSE
TimesSent	int	11	FALSE	FALSE	FALSE
Content	longtext	0	FALSE	FALSE	FALSE
ActivityId	longtext	0	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE

RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 33 - Table IntegrationEvents

2.3.7 Table News

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Title	varchar	200	FALSE	FALSE	FALSE
UserId	char	36	FALSE	FALSE	FALSE
PublishedAt	datetime	6	FALSE	TRUE	FALSE
Category	varchar	50	FALSE	FALSE	FALSE
Content	longtext	0	FALSE	FALSE	FALSE
ImageUrl	varchar	500	FALSE	FALSE	FALSE
Status	varchar	50	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 34 - Table News

2.3.8 Table Orders

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
EstimatedTotal	decimal	18	FALSE	FALSE	FALSE
PaymentStatus	int	10	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 35 - Table News

2.3.9 Table OrderItems

Name	Type	Length	Primary Key	Nullable	Unique

Id	char	36	TRUE	FALSE	TRUE
OrderId	char	36	FALSE	FALSE	FALSE
PackageId	char	36	FALSE	FALSE	FALSE
PackageName	varchar	255	FALSE	FALSE	FALSE
Price	decimal	18	FALSE	FALSE	FALSE
Discount	decimal	18	FALSE	FALSE	FALSE
PriceAfterDiscount	decimal	18	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
Rowversion	timestamp	6	FALSE	TRUE	FALSE

2.3.10 Table Wishlists

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
Name	varchar	100	FALSE	FALSE	FALSE
Description	varchar	500	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 36 - Table OrderItems

2.3.11 Table WishlistItems

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
WishlistId	char	36	FALSE	FALSE	FALSE
PackageId	char	36	FALSE	TRUE	FALSE
Note	varchar	255	FALSE	FALSE	FALSE
Priority	int	10	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE

UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	varchar	255	FALSE	TRUE	FALSE

Table 37 - Table OrderItems

2.3.12 Table Libraries

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 38 - Table Libraries

2.3.13 Table LibraryItems

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
LibraryId	char	36	FALSE	FALSE	FALSE
PackageId	char	36	FALSE	FALSE	FALSE
AddedTime	datetime	6	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	36	FALSE	TRUE	FALSE
UpdatedBy	varchar	36	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 39 - Table LibraryItems

2.3.14 Table Projects

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
ProjectKey	char	36	FALSE	FALSE	FALSE

ProjectName	varchar	150	FALSE	FALSE	FALSE
Status	int	10	FALSE	FALSE	FALSE
UserId	char	36	FALSE	FALSE	FALSE
GamePackageId	char	36	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 40 - Table Projects

2.3.15 Table Players

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Name	varchar	100	FALSE	FALSE	FALSE
Account	varchar	100	FALSE	FALSE	FALSE
Password	varchar	100	FALSE	FALSE	FALSE
Status	int	10	FALSE	FALSE	FALSE
ProjectId	char	36	FALSE	FALSE	FALSE
UserId	char	36	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 41 - Table Players

2.3.16 Table PlayerLogs

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
CheckInTime	date	6	FALSE	FALSE	FALSE
CheckOutTime	date	6	FALSE	FALSE	FALSE
PlayerId	char	36	FALSE	FALSE	FALSE

CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 42 - Table PlayerLogs

2.3.17 Table PlayerGameItems

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
PlayerId	char	36	FALSE	FALSE	FALSE
GameItemId	char	36	FALSE	FALSE	FALSE
Quantity	int	10	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 43 - Table PlayerGameItems

2.3.18 Table WorkflowProcesses

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
ProcessCode	char	100	FALSE	FALSE	TRUE
Name	varchar	300	FALSE	FALSE	FALSE
Description	varchar	1500	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 44 - Table WorkflowProcesses

2.3.19 Table WorkflowRequests

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
WorkflowProcessId	char	36	FALSE	FALSE	FALSE
Title	varchar	500	FALSE	TRUE	FALSE
CurrentStateId	int	11	FALSE	FALSE	FALSE
CurrentStage	int	11	FALSE	FALSE	FALSE
EntityCodeOnTrack	char	100	FALSE	FALSE	FALSE
EntityKeyId	varchar	255	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 45 - Table WorkflowRequests

2.3.20 Table WorkflowRequestOnTrackHistories

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
ActionTriggerId	char	36	FALSE	FALSE	FALSE
Stage	int	11	FALSE	FALSE	FALSE
ProcessStateId	int	11	FALSE	FALSE	FALSE
WorkflowRequestId	char	36	FALSE	FALSE	FALSE
ActorTriggerId	char	36	FALSE	FALSE	FALSE
Comments	longtext	0	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 46 - Table WorkflowRequestOnTrackHistories

2.3.21 Table Transitions

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
WorkFlowProcessId	char	36	FALSE	FALSE	FALSE
CurrentStateId	int	11	FALSE	FALSE	FALSE
NextStateId	int	11	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 47 - Transitions

2.3.22 Table ActionTriggers

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Name	varchar	255	FALSE	FALSE	FALSE
Description	varchar	1500	FALSE	TRUE	FALSE
ActionTriggerType	int	11	FALSE	FALSE	FALSE
WorkflowProcessId	char	36	FALSE	FALSE	FALSE
StepOrderNumber	int	11	FALSE	FALSE	FALSE
TransitionId	char	36	FALSE	FALSE	FALSE
IsAllowBackStep	tinyint	1	FALSE	FALSE	FALSE
AllowBackToStepOrderNumber	int	11	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 48 - Table ActionTriggers

2.3.23 Table ProcessUsers

Name	Type	Length	Primary Key	Nullable	Unique

Id	char	36	TRUE	FALSE	TRUE
WorkflowProcessId	char	36	FALSE	FALSE	FALSE
ActionTriggerId	char	36	FALSE	FALSE	FALSE
UserGroupIdHasAuthorities	char	36	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 49 - Table ProcessUsers

2.3.24 Table ProcessStates

Name	Type	Length	Primary Key	Nullable	Unique
Id	int	11	TRUE	FALSE	TRUE
Name	varchar	255	FALSE	FALSE	FALSE
Description	varchar	1500	FALSE	TRUE	FALSE
StateType	int	11	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 50 - Table ProcessStates

2.3.25 Table UserGroups

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
GroupName	varchar	255	FALSE	FALSE	TRUE
BasedRoles	longtext	0	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 51 - Table UserGroups

2.3.26 Table SupportTickets

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Title	varchar	200	FALSE	FALSE	FALSE
Description	text	0	FALSE	FALSE	FALSE
UserId	char	36	FALSE	FALSE	FALSE
Status	int	11	FALSE	FALSE	FALSE
Category	varchar	100	FALSE	TRUE	FALSE
AttachmentsKey	varchar	500	FALSE	TRUE	FALSE
AttachmentsFileName	varchar	500	FALSE	TRUE	FALSE
ResolvedAt	datetime	6	FALSE	TRUE	FALSE
ResolvedBy	varchar	255	FALSE	TRUE	FALSE
ResponseBody	text	0	FALSE	TRUE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
CreatedBy	varchar	255	FALSE	TRUE	FALSE
UpdatedBy	varchar	255	FALSE	TRUE	FALSE
RowVersion	timestamp	6	FALSE	TRUE	FALSE

Table 52 - Table SupportTickets

2.3.27 Table Coins

Name	Type	Length	Primary Key	Nullable	Unique
CoinId	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
Amount	double	0	FALSE	TRUE	FALSE

Table 53 - Table Coins

2.3.28 Table TransactionCoins

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
AuthorId	char	36	FALSE	FALSE	FALSE

PackageId	char	36	FALSE	FALSE	FALSE
PackageName	varchar	1000	FALSE	FALSE	FALSE
Amount	double	0	FALSE	FALSE	FALSE
Description	varchar	1500	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
OrderId	char	36	FALSE	FALSE	FALSE
Type	int	11	FALSE	FALSE	FALSE
OriginalPrice	double	0	FALSE	FALSE	FALSE

Table 54 - Table TransactionCoins

2.3.29 Table Transactions

Name	Type	Length	Primary Key	Nullable	Unique
TransactionId	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
Amount	int	11	FALSE	FALSE	FALSE
OrderCode	bigint	20	FALSE	FALSE	FALSE
Status	varchar	20	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
Type	int	11	FALSE	FALSE	FALSE

Table 55 - Table Transactions

2.3.30 Table SystemWallet

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	double	0	FALSE	FALSE	FALSE

Table 56 - Table SystemWallet

2.3.31 Table WithdrawalRequests

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
Amount	double	0	FALSE	FALSE	FALSE
BankAccountNumber	longtext	0	FALSE	FALSE	FALSE
BankName	longtext	0	FALSE	FALSE	FALSE

AccountHolderName	longtext	0	FALSE	FALSE	FALSE
Status	int	11	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE
UpdatedAt	datetime	6	FALSE	TRUE	FALSE
RejectReason	longtext	0	FALSE	TRUE	FALSE
Note	longtext	0	FALSE	TRUE	FALSE
ReceiptKey	longtext	0	FALSE	TRUE	FALSE
ReceiptFileName	longtext	0	FALSE	TRUE	FALSE

Table 57 - Table WithdrawalRequests

2.3.32 Table RevenueShares

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Percentage	double	0	FALSE	FALSE	FALSE
CreatedAt	datetime	6	FALSE	FALSE	FALSE

Table 58 - Table RevenueShares

2.3.33 Table Roles

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Name	varchar	100	FALSE	FALSE	FALSE
Description	varchar	1500	FALSE	TRUE	FALSE

Table 59 - Table Roles

2.3.34 Table UserRoles

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
UserId	char	36	FALSE	FALSE	FALSE
RoleId	char	36	FALSE	FALSE	FALSE

Table 60 - Table UserRoles

2.3.35 Table RolePermissions

Name	Type	Length	Primary Key	Nullable	Unique

Id	char	36	TRUE	FALSE	TRUE
RoleId	char	36	FALSE	FALSE	FALSE
PermissionId	char	36	FALSE	FALSE	FALSE

Table 61 - Table RolePermissions

2.3.36 Table Permissions

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
Name	varchar	200	FALSE	FALSE	FALSE
Description	varchar	1500	FALSE	TRUE	FALSE

Table 62 - Table Permissions

2.3.37 Table ApiPermissions

Name	Type	Length	Primary Key	Nullable	Unique
Id	char	36	TRUE	FALSE	TRUE
ApiPath	varchar	255	FALSE	FALSE	FALSE
PermissionId	char	36	FALSE	FALSE	FALSE
ApiMethod	char	50	FALSE	FALSE	FALSE

Table 63 - Table ApiPermissions

2.3.38 Collection Game Package

Name	Type	Primary Key	Required
Id	binary	TRUE	FALSE
Name	string	FALSE	FALSE
ProductSaleType	numeric	FALSE	FALSE
GamePackageType	numeric	FALSE	FALSE
GameProductId	binary	FALSE	TRUE
CurrentPrice	document	FALSE	TRUE
CurrentPrice.Price	numeric	FALSE	FALSE
CurrentPrice.DiscountedPrice	numeric	FALSE	FALSE
CurrentPrice.Currency	string	FALSE	FALSE

CurrentDiscount	document	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages	document	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.DiscountId	binary	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.DiscountPackageId	binary	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.StartDate	array	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.StartDate.[0]	numeric	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.EndDate	array	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.EndDate.[0]	numeric	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.Type	numeric	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.DiscountName	string	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.TotalDiscount	numeric	FALSE	TRUE
CurrentDiscount.Type	numeric	FALSE	TRUE
CurrentDiscount.Value	numeric	FALSE	TRUE
PackageReleaseStatus	numeric	FALSE	FALSE
ReleaseDate	array	FALSE	FALSE
ReleaseDate.[0]	numeric	FALSE	FALSE
NumericId	numeric	FALSE	TRUE
Slug	string	FALSE	FALSE
SupportedPlatforms	document	FALSE	FALSE
SupportedPlatforms.PlatformTypeId	binary	FALSE	TRUE
SupportedPlatforms.Minimum	document	FALSE	TRUE
SupportedPlatforms.Minimum.AdditionalNotes	string	FALSE	FALSE

SupportedPlatforms.Minimum.DirectX	string	FALSE	TRUE
SupportedPlatforms.Minimum.Graphics	string	FALSE	TRUE
SupportedPlatforms.Minimum.Memory	string	FALSE	TRUE
SupportedPlatforms.Minimum.OSVersion	string	FALSE	TRUE
SupportedPlatforms.Minimum.Processor	string	FALSE	TRUE
SupportedPlatforms.Minimum.Storage	string	FALSE	TRUE
SupportedPlatforms.Recommend	document	FALSE	FALSE
SupportedPlatforms.Recommend.DirectX	string	FALSE	TRUE
SupportedPlatforms.Recommend.Graphics	string	FALSE	TRUE
SupportedPlatforms.Recommend.Memory	string	FALSE	TRUE
SupportedPlatforms.Recommend.OSVersion	string	FALSE	TRUE
SupportedPlatforms.Recommend.Processor	string	FALSE	TRUE
SupportedPlatforms.Recommend.Storage	string	FALSE	TRUE
SourceFilesGame	document	FALSE	FALSE
SourceFilesGame.FileName	string	FALSE	FALSE
SourceFilesGame.Key	string	FALSE	TRUE
SourceFilesGame.OS	string	FALSE	TRUE
SourceFilesGame.SourceName	string	FALSE	TRUE
ShortDescription	string	FALSE	FALSE
LongDescription	string	FALSE	FALSE
Media	document	FALSE	FALSE
Media.MainImage	document	FALSE	FALSE
Media.MainImage.Title	string	FALSE	FALSE
Media.MainImage.MediaId	string	FALSE	FALSE
Media.MainImage.MediaType	numeric	FALSE	TRUE
Media.MainImage.Image	document	FALSE	TRUE

Media.MainImage.Image.AltText	string	FALSE	FALSE
Media.MainImage.Image.Format	document	FALSE	FALSE
Media.MainImage.Image.Url	document	FALSE	TRUE
Media.MainVideo	document	FALSE	FALSE
Media.MainVideo.Title	string	FALSE	FALSE
Media.MainVideo.MediaId	string	FALSE	FALSE
Media.MainVideo.MediaType	numeric	FALSE	TRUE
Media.MainVideo.Files	array	FALSE	TRUE
Media.MainVideo.Files.[0]	document	FALSE	FALSE
Media.MainVideo.Files.[0].AltText	string	FALSE	FALSE
Media.MainVideo.Files.[0].Format	document	FALSE	FALSE
Media.MainVideo.Files.[0].Url	document	FALSE	TRUE
Media.MainVideo.StillFrameMediaId	string	FALSE	FALSE
Media.MediaItems	array	FALSE	FALSE
Media.MediaItems.[0]	document	FALSE	FALSE
Media.MediaItems.[0].Title	string	FALSE	FALSE
Media.MediaItems.[0].MediaId	string	FALSE	FALSE
Media.MediaItems.[0].MediaType	numeric	FALSE	TRUE
Media.MediaItems.[0].Video	document	FALSE	FALSE
Media.MediaItems.[0].Video.Files	array	FALSE	TRUE
Media.MediaItems.[0].Video.Files.[0]	document	FALSE	FALSE
Media.MediaItems.[0].Video.Files.[0].AltText	string	FALSE	FALSE
Media.MediaItems.[0].Video.Files.[0].Format	document	FALSE	FALSE
Media.MediaItems.[0].Video.Files.[0].Url	document	FALSE	TRUE
Media.MediaItems.[0].Video.StillFrameMediaId	string	FALSE	FALSE
Media.MediaItems.[0].Image	document	FALSE	TRUE

Media.MediaItems.[0].Image.AltText	string	FALSE	FALSE
Media.MediaItems.[0].Image.Format	document	FALSE	FALSE
Media.MediaItems.[0].Image.Url	document	FALSE	TRUE
Rating	numeric	FALSE	FALSE
TotalReviews	numeric	FALSE	FALSE
TotalSale	numeric	FALSE	FALSE
TotalViewers	numeric	FALSE	FALSE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 64 - Collection Game Package

2.3.39 Collection Game Product

Name	Type	Primary Key	Required
Id	binary	TRUE	TRUE
Name	string	FALSE	TRUE
Developer	string	FALSE	TRUE
Publisher	string	FALSE	TRUE
AuthorId	binary	FALSE	TRUE
ProductType	numeric	FALSE	TRUE
AssetProductCategory	document	FALSE	FALSE
AssetProductCategory.AssetProductTypeId	binary	FALSE	TRUE
AssetProductCategory.Values	array	FALSE	TRUE
AssetProductCategory.Values.[0]	string	FALSE	FALSE
Categories	array	FALSE	FALSE
Categories.[0]	document	FALSE	FALSE

Categories.[0].GenreTypeId	binary	FALSE	TRUE
Categories.[0].Values	array	FALSE	TRUE
Categories.[0].Values.[0]	string	FALSE	FALSE
PlayerModes	array	FALSE	FALSE
PlayerModes.[0]	document	FALSE	FALSE
PlayerModes.[0].PlayerModeTypeId	binary	FALSE	TRUE
SocialMedias	array	FALSE	FALSE
SocialMedias.[0]	document	FALSE	FALSE
SocialMedias.[0].SocialMediaTypeId	binary	FALSE	TRUE
SocialMedias.[0].Url	string	FALSE	TRUE
ExternalLinks	array	FALSE	FALSE
ExternalLinks.[0]	document	FALSE	FALSE
ExternalLinks.[0].LinkName	string	FALSE	TRUE
ExternalLinks.[0].LinkUrl	string	FALSE	TRUE
GameRatings	array	FALSE	FALSE
GameRatings.[0]	document	FALSE	FALSE
GameRatings.[0].GameContentRatingTypeId	binary	FALSE	TRUE
GameRatings.[0].Desciptors	string	FALSE	TRUE
SupportedLanaguages	array	FALSE	FALSE
SupportedLanaguages.[0]	string	FALSE	FALSE
LegalLines	string	FALSE	FALSE
SupportContactInfo	document	FALSE	TRUE
SupportContactInfo.SupportEmail	string	FALSE	TRUE
SupportContactInfo.SupportPhone	string	FALSE	FALSE
SupportContactInfo.SupportWebsiteUrl	string	FALSE	TRUE
Tags	array	FALSE	FALSE
Tags.[0]	string	FALSE	FALSE
SearchKeywords	array	FALSE	FALSE
SearchKeywords.[0]	string	FALSE	FALSE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE

UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 65 - Collection Game Product

2.3.40 Collection NotificationSystems

Name	Type	Primary Key	Required
Id	binary	TRUE	TRUE
UserSendId	binary	FALSE	TRUE
UserRecipientId	binary	FALSE	FALSE
UserRecipientEmail	string	FALSE	FALSE
UseCaseNotiType	numeric	FALSE	TRUE
Content	document	FALSE	FALSE
Content.Title	string	FALSE	TRUE
Content.OnClickUrl	string	FALSE	FALSE
Content.Description	string	FALSE	TRUE
Content.CoverImageLink	string	FALSE	FALSE
ChannelTracking	array	FALSE	FALSE
ChannelTracking.[0]	document	FALSE	FALSE
ChannelTracking.[0].Channel	numeric	FALSE	TRUE
ChannelTracking.[0].IsRead	boolean	FALSE	TRUE
ChannelTracking.[0].SentAt	array	FALSE	FALSE
ChannelTracking.[0].SentAt.[0]	numeric	FALSE	FALSE
ChannelTracking.[0].StatusSending	numeric	FALSE	TRUE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 66 - NotificationSystems

2.3.41 Collection Review Request

Name	Type	Primary Key	Required
Id	binary	TRUE	TRUE
AssetPackageId	binary	FALSE	FALSE
GamePackageId	binary	FALSE	FALSE
IsApproved	boolean	FALSE	TRUE
ObjectEntityCode	string	FALSE	TRUE
ProcessCodeInUsed	string	FALSE	TRUE
ReviewRequestTypeName	string	FALSE	TRUE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 67 - Collection Review Request

2.3.42 Collection System Category

Name	Type	Primary Key	Required
Id	binary	TRUE	TRUE
CategoryTypeName	string	FALSE	TRUE
CategoryLabelName	string	FALSE	TRUE
SubAttributes	document	FALSE	FALSE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 68 - Collection System Category

2.3.43 Collection Game Item

Name	Type	Primary key	Required
Id	binary	TRUE	TRUE
Name	string	FALSE	TRUE
Type	string	FALSE	TRUE
Description	string	FALSE	TRUE
Price	numeric	FALSE	FALSE
IsStackable	boolean	FALSE	TRUE
Rarity	string	FALSE	TRUE
Image	document	FALSE	TRUE
ProjectId	binary	FALSE	TRUE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 69 - Collection Game Item

2.3.44 Collection Feedback Package

Name	Type	Primary key	Required
Id	binary	TRUE	TRUE
GamePackageId	binary	FALSE	TRUE
isRecommenned	boolean	FALSE	TRUE
Rating	numeric	FALSE	TRUE
UserFeedbackId	binary	FALSE	TRUE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE

UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 70 - Collection Feedback Package

2.3.45 Collection Asset Package

Name	Type	Primary key	Required
Id	binary	TRUE	TRUE
Name	string	FALSE	FALSE
ProductSaleType	numeric	FALSE	FALSE
GamePackageType	numeric	FALSE	FALSE
GameProductId	binary	FALSE	TRUE
CurrentPrice	document	FALSE	TRUE
CurrentPrice.Price	numeric	FALSE	FALSE
CurrentPrice.DiscountedPrice	numeric	FALSE	FALSE
CurrentPrice.Currency	string	FALSE	FALSE
CurrentDiscount	document	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages	document	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.DiscountId	binary	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.DiscountPackageId	binary	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.StartDate	array	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.StartDate.[0]	numeric	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.EndDate	array	FALSE	FALSE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.EndDate.[0]	numeric	FALSE	FALSE

CurrentDiscount.CurrentActivatedAppliedDiscountPackages.Type	numeric	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.DiscountName	string	FALSE	TRUE
CurrentDiscount.CurrentActivatedAppliedDiscountPackages.TotalDiscount	numeric	FALSE	TRUE
CurrentDiscount.Type	numeric	FALSE	TRUE
CurrentDiscount.Value	numeric	FALSE	TRUE
PackageReleaseStatus	numeric	FALSE	FALSE
ReleaseDate	array	FALSE	FALSE
ReleaseDate.[0]	numeric	FALSE	FALSE
NumericId	numeric	FALSE	TRUE
Slug	string	FALSE	FALSE
SourceFileAssets	document	FALSE	FALSE
SourceFileAssets.GroupFormat	string	FALSE	FALSE
SourceFileAssets.SourceFiles	array	FALSE	FALSE
SourceFileAssets.SourceFiles.[0]	document	FALSE	FALSE
SourceFileAssets.SourceFiles.[0].FileName	string	FALSE	FALSE
SourceFileAssets.SourceFiles.[0].Key	string	FALSE	TRUE
SourceFileAssets.SourceFiles.[0].SourceName	string	FALSE	TRUE
ShortDescription	string	FALSE	FALSE
LongDescription	string	FALSE	FALSE
Media	document	FALSE	FALSE
Media.MainImage	document	FALSE	FALSE
Media.MainImage.Title	string	FALSE	FALSE
Media.MainImage.MediaId	string	FALSE	FALSE
Media.MainImage.MediaType	numeric	FALSE	TRUE
Media.MainImage.Image	document	FALSE	TRUE

Media.MainImage.Image.AltText	string	FALSE	FALSE
Media.MainImage.Image.Format	document	FALSE	FALSE
Media.MainImage.Image.Url	document	FALSE	TRUE
Media.MainVideo	document	FALSE	FALSE
Media.MainVideo.Title	string	FALSE	FALSE
Media.MainVideo.MediaId	string	FALSE	FALSE
Media.MainVideo.MediaType	numeric	FALSE	TRUE
Media.MainVideo.Files	array	FALSE	TRUE
Media.MainVideo.Files.[0]	document	FALSE	FALSE
Media.MainVideo.Files.[0].AltText	string	FALSE	FALSE
Media.MainVideo.Files.[0].Format	document	FALSE	FALSE
Media.MainVideo.Files.[0].Url	document	FALSE	TRUE
Media.MainVideo.StillFrameMediaId	string	FALSE	FALSE
Media.MediaItems	array	FALSE	FALSE
Media.MediaItems.[0]	document	FALSE	FALSE
Media.MediaItems.[0].Title	string	FALSE	FALSE
Media.MediaItems.[0].MediaId	string	FALSE	FALSE
Media.MediaItems.[0].MediaType	numeric	FALSE	TRUE
Media.MediaItems.[0].Video	document	FALSE	FALSE
Media.MediaItems.[0].Video.Files	array	FALSE	TRUE
Media.MediaItems.[0].Video.Files.[0]	document	FALSE	FALSE
Media.MediaItems.[0].Video.Files.[0].AltText	string	FALSE	FALSE
Media.MediaItems.[0].Video.Files.[0].Format	document	FALSE	FALSE
Media.MediaItems.[0].Video.Files.[0].Url	document	FALSE	TRUE
Media.MediaItems.[0].Video.StillFrameMediaId	string	FALSE	FALSE
Media.MediaItems.[0].Image	document	FALSE	TRUE

Media.MediaItems.[0].Image.AltText	string	FALSE	FALSE
Media.MediaItems.[0].Image.Format	document	FALSE	FALSE
Media.MediaItems.[0].Image.Url	document	FALSE	TRUE
Rating	numeric	FALSE	FALSE
TotalReviews	numeric	FALSE	FALSE
TotalSale	numeric	FALSE	FALSE
TotalViewers	numeric	FALSE	FALSE
CreatedAt	array	FALSE	TRUE
CreatedAt.[0]	numeric	FALSE	FALSE
UpdatedAt	array	FALSE	FALSE
UpdatedAt.[0]	numeric	FALSE	FALSE
CreatedBy	string	FALSE	FALSE
UpdatedBy	string	FALSE	FALSE
RowVersion	string	FALSE	FALSE

Table 71 - Collection Asset Package

3. Detailed Design

3.1 Authentication & Authorization

3.1.1 Class Diagram

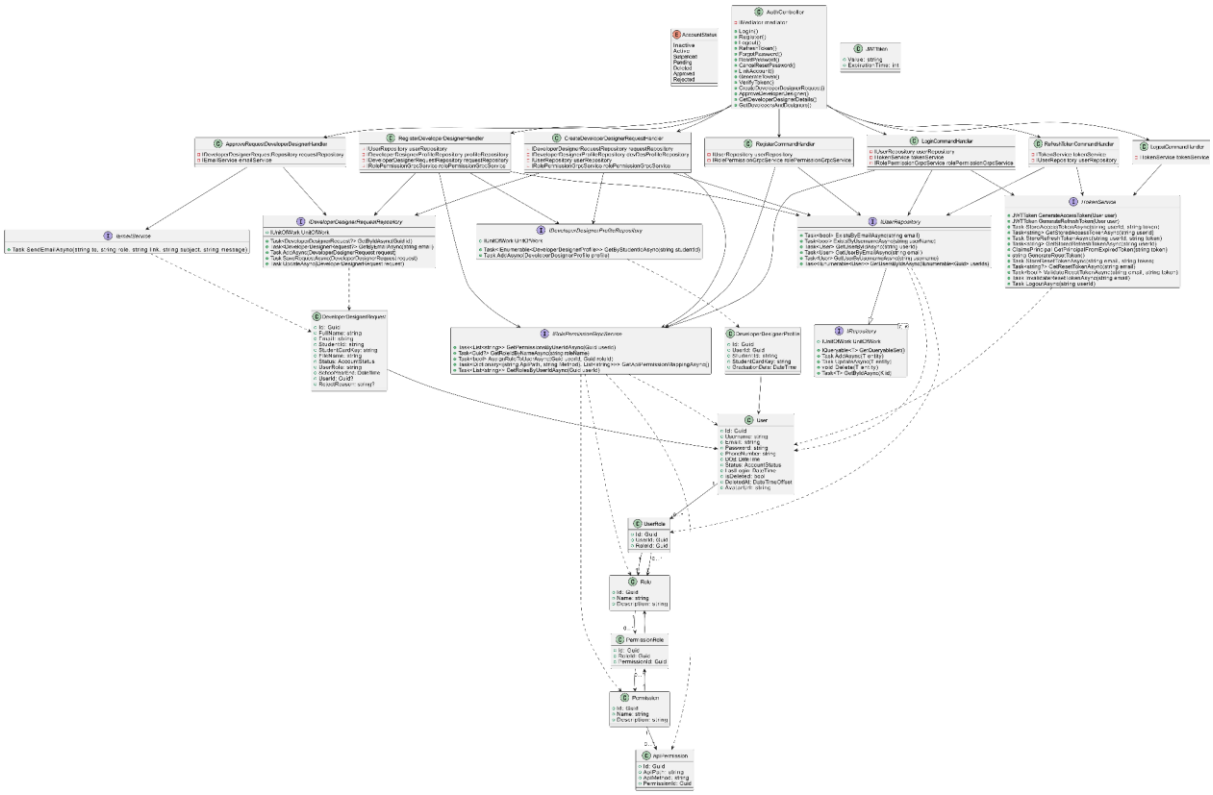

Figure 114 - Authentication & Authorization Class Diagram

Authentication & Authorization

3.1.2 Class Diagram Specification

3.1.2.1 Auth Controller

No	Method	Description
1	Login()	Handles the user login process.
2	Register()	Registers a new user.
3	Logout()	Logs the user out and clears the session/token.
4	RefreshToken()	Issues a new JWT token using the refresh token.
5	ForgotPassword()	Initiates the password reset process by sending a reset token to the user's email.
6	ResetPassword()	Resets the user’s password using the provided reset token.

7	CancelResetPassword()	Cancels an ongoing password reset request.
8	LinkAccount()	Links the current user account with an external or additional identity.
9	GenerateToken()	Generates a token manually for validate
10	VerifyToken()	Validate generated token.
11	CreateDeveloperDesignerRequest()	Submits a request to be recognized as a developer or designer.
12	ApproveDeveloperDesigner()	Approves a request to become a developer or designer.
13	GetDeveloperDesignerDetails()	Retrieves details about the developer or designer role for the user.
14	GetDevelopersAndDesigners()	Returns a list of users with developer or designer roles.

3.1.2.2 Auth Command Handlers

No	Method	Description
1	LoginCommandHandler	Handles user login: verifies credentials, generates JWT tokens, assigns permissions.
2	RegisterCommandHandler	Handles new user registration and initial role assignment.
3	LogoutCommandHandler	Handles user logout by invalidating stored refresh token.
4	RefreshTokenCommandHandler	Validates a refresh token and issues a new access token.
5	CreateDeveloperDesignerRequestHandler	Validates and creates a request to register as developer or designer.
6	ApproveRequestDeveloperDesignerHandler	Approves or rejects a developer/designer registration request and sends notification email.
7	RegisterDeveloperDesignerHandler	Completes developer/designer registration: creates User, assigns role, creates Profile, adds subscription.

3.1.2.3 Auth Service

No	Method	Description

1	ITokenService	Issues, stores, validates, and revokes JWT access and refresh tokens; handles password reset tokens and logout.
2	IRolePermissionGrpcService	Retrieves roles and permissions mappings via gRPC.
3	IEmailService	Sends email notifications (to address, role context, link, subject, body).

3.1.2.4 Auth Repository

No	Method	Description
1	IRepository<T,Key>	Generic CRUD operations and unit-of-work support.
2	IUserRepository	User-specific queries (exists by email/username, fetch by id/email/username).
3	IDeveloperDesignerRequestRepository	CRUD and custom operations for DeveloperDesignerRequest.
4	IDeveloperDesignerProfileRepository	Queries and persistence for DeveloperDesignerProfile.

3.1.2.5 Entity

No	Method	Description
1	User	Stores user information including credentials, profile data, and status.
2	UserRole	Junction entity linking users to roles (many-to-many relationship).
3	Role	Defines a named role grouping a set of permissions.
4	Permission	Defines an individual permission or privilege.
5	RolePermission	Junction entity linking roles to permissions.
6	ApiPermission	Maps an API endpoint (path + method) to a required permission.
7	JWTToken	Holds a JWT value and its expiration time.
8	DeveloperDesignerProfile	Profile data for a developer or designer (student ID, gradua
9	DeveloperDesignerRequest	A request to register as developer/designer, with status, role, and documents.

3.1.3 Sequence Diagram: Register

For Customer:

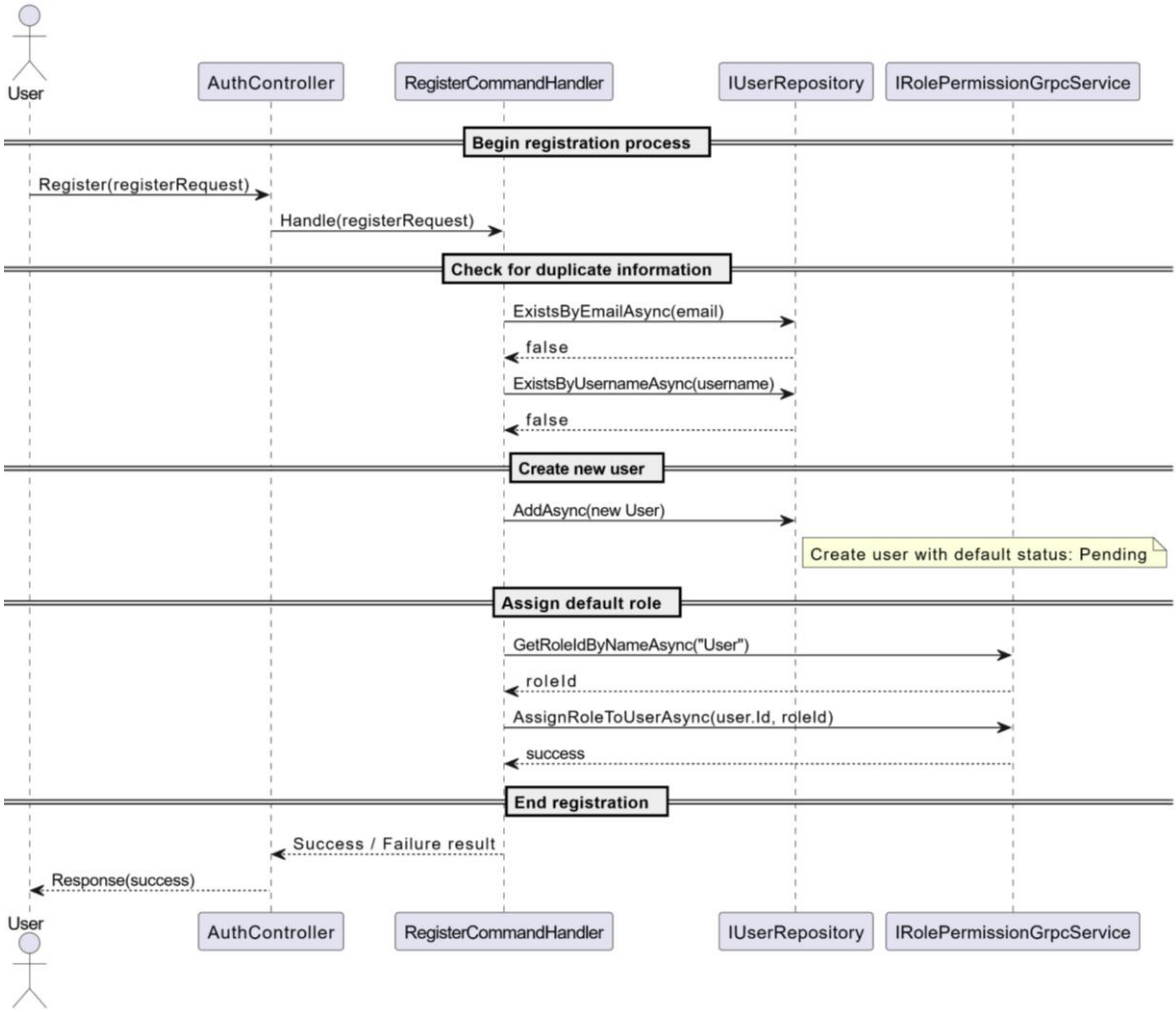

Figure 115 - Register For Customer Sequence Diagram

For Developer, Designer:

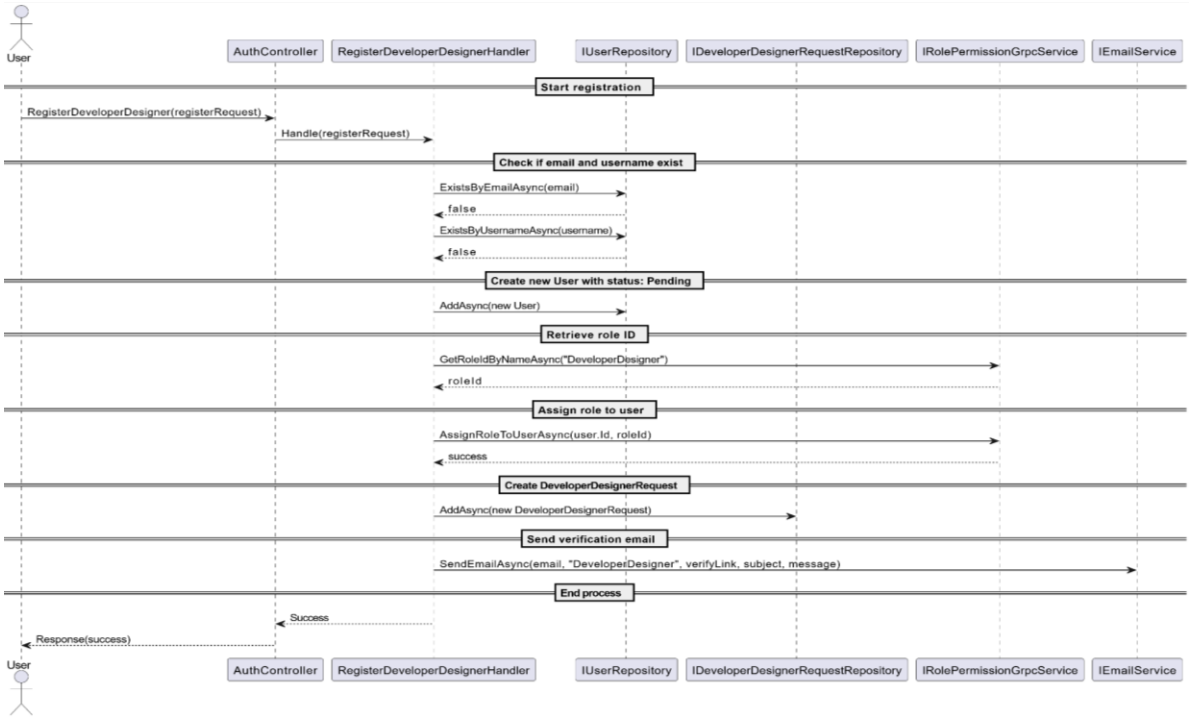

Figure 116 - Register For Developer, Designer Sequence Diagram

3.1.4 Sequence Diagram: Login

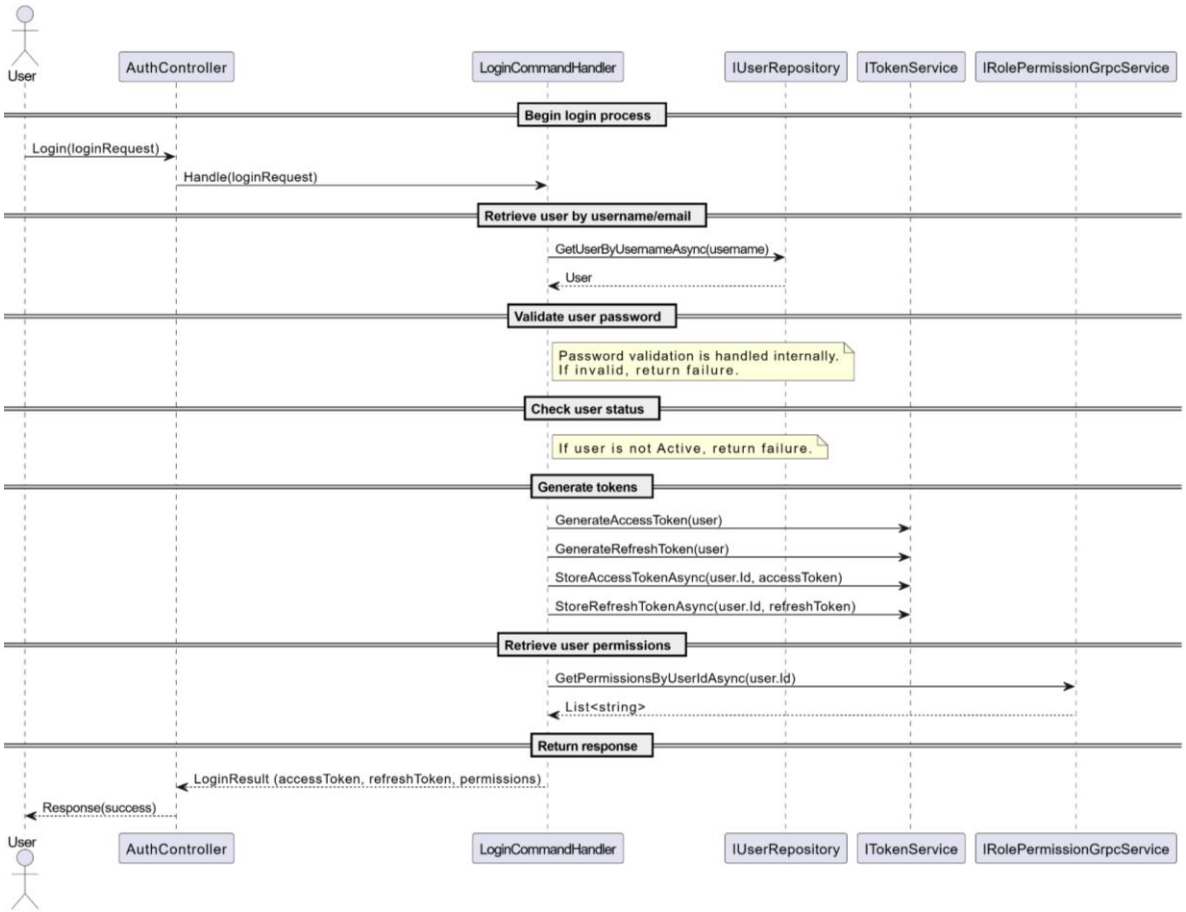

Figure 117 – Login Sequence Diagram

3.2 Purchase Game Product & Game Asset

3.2.1 Class Diagram

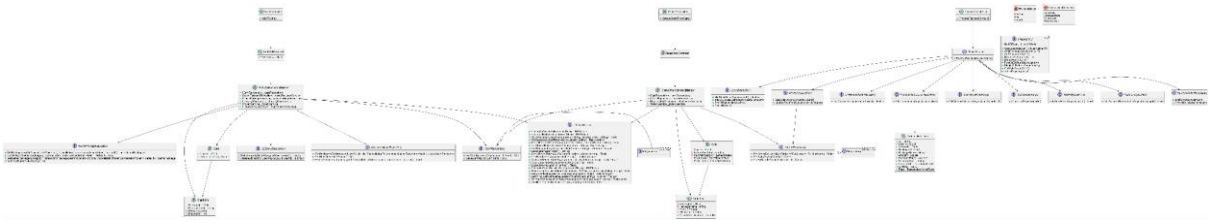

Figure 118 – Payment Class Diagram

Payment Class Diagram

3.2.2 Class Diagram Specification

3.2.2.1 CartController

No	Method	Description
1	AddToCart()	Handles HTTP request to add a package to the user's cart.

3.2.2.2 OrderController

No	Method	Description
1	CreateOrderFromCart()	Creates an order based on the user's current cart.

3.2.2.3 PaymentController

No	Method	Description
1	ProcessPayment(orderId)	Handles the payment logic and updates related transaction information.

3.2.2.4 CartRepository

No	Method	Description
1	GetCartByUserIdAsync(Guid)	Gets cart data for a specific user.
2	SaveCartAsync(Cart)	Persists or updates cart information.

3.2.2.5 OrderRepository

No	Method	Description
1	GetOrderByUserIdAndOrderId	Gets a specific order based on user ID and order ID.

	(Guid, Guid)	
2	GetbyIdAsync(Guid)	Get order by ID.
3	GetByUserIdAsync(Guid)	Gets all orders of a user.

3.2.2.6 LibraryRepository

No	Method	Description
1	GetLibraryByUserIdAsync(Guid)	Retrieves game library data of a user.
2	CreateLibraryForUserAsync(Guid)	Creates a game library record for a new user.

3.2.2.7 AssetPackageRepository

No	Method	Description
1	GetPackagesByIdsAsync(List<Guid>, CancellationToken)	Retrieves asset packages by ID list.
2	GetNextNumericIdAsync()	Gets the next available numeric ID for asset package.
3	ExistsAsync(Guid, CancellationToken)	Checks whether a package with the given ID exists.

3.2.2.8 GamePackageRepository

No	Method	Description
1	GetPackagesByIdsAsync(List<Guid>, CancellationToken)	Gets game packages by list of IDs.
2	ExistsAsync(Guid, CancellationToken)	Verifies if the package exists.
3	GetGamePackagesListApplyFiltersAsync(PackageSearchOptions, CancellationToken)	Retrieves game packages based on filter criteria.
4	GetNextNumericIdAsync()	Gets the next numeric ID for game packages.

3.2.2.9 TokenService

No	Method	Description
1	GenerateAccessToken(User)	Generates access JWT token.
2	GenerateRefreshToken(User)	Generates refresh JWT token.

3	StoreAccessTokenAsync(string, string)	Stores access token by user ID.
4	GetStoredAccessTokenAsync(string)	Retrieves stored access token.
5	StoreRefreshTokenAsync(string, string)	Stores refresh token.
6	GetStoredRefreshTokenAsync(string)	Gets refresh token from storage.
7	GetPrincipalFromExpiredToken(string)	Gets principal identity from an expired token.
8	GenerateResetToken()	Generates a password reset token.
9	StoreResetTokenAsync(string, string)	Stores password reset token by email.
10	GetResetTokenAsync(string)	Retrieves reset token from storage.
11	ValidateResetTokenAsync(string, string)	Validates the reset token.
12	InvalidateResetTokenAsync(string)	Invalidates a stored reset token.
13	LogoutAsync(string)	Logs out the user and clears the session.
14	GenerateGameSessionToken(Player)	Generates a token for a game session.
15	StoreGameSessionTokenAsync(Guid, string)	Stores a game session token.
16	GetStoredGameSessionTokenAsync(Guid)	Retrieves stored game session token.
17	GetPrincipalFromGameSessionToken(string)	Extracts principal from game session token.
18	ValidateGameSessionTokenAsync(Guid, string)	Validates a stored game session token.
19	InvalidateGameSessionTokenAsync(Guid)	Invalidates a game session token.

3.2.2.10 CoinService

No	Method	Description
1	ProcessPaymentAsync(Guid orderId)	Processes payment and handles coins.

3.2.2.11 OrderServiceGrpc

No	Method	Description
1	GetOrderAsync(orderId, userId)	Gets order via gRPC.
2	UpdatePaymentStatus(orderId, status)	Updates payment status using gRPC.

3.2.2.12 LibraryServiceGrpc

No	Method	Description
1	AddGamesToLibraryAsync(Guid, Guid)	Adds purchased games to user’s library.

3.2.2.13 CartServiceGrpc

No	Method	Description
1	ClearCartAsync(Guid)	Clears cart data after checkout.

3.2.2.14 UserGrpcService

No	Method	Description
1	IsUserExistAsync(Guid)	Checks if a user exists by ID.

3.2.2.15 AuthorGrpcClient

No	Method	Description
1	GetAuthorIdByProductIdAsync(Guid)	Gets the author ID for a given package ID.

3.2.2.16 SystemWalletRepository

No	Method	Description
1	AddCoinsAsync(double)	Adds coins to the system wallet.
2	DeductCoinsAsync(double)	Deducts coins from the system wallet.

3.2.2.17 CoinRepository

No	Method	Description
1	GetUserCoinsAsync(Guid)	Retrieves a user's current coin balance.
2	AddOrUpdateCoinsAsync(Guid, double)	Updates coin balance.
3	BeginTransactionAsync()	Begins a database transaction.
4	CommitAsync()	Commits a transaction.

3.2.2.18 TransactionCoinsRepository

No	Method	Description
1	AddTransactionAsync(object)	Adds a new transaction to the coin history.

3.2.2.19 RevenueShareRepository

No	Method	Description
1	GetCurrentRevenueShareAsync()	Retrieves the current revenue share percentage.

3.2.2.20 Cart

No	Property	Description
1	UserId	ID of user
2	Items	List of item in cart

3.2.2.21 CartItem

No	Property	Description
1	PackageId	Id of package
2	PackageName	Name of package
3	Price	Price of package
4	Discount	Discount of package

3.2.2.22 Order

No	Property	Description
1	UserId	ID of user
2	EstimatedTotal	Total price
3	PaymentStatus	Status of order
4	CreatedAt	Date created of order
5	OrderItems	List item in order

3.2.2.23 OrderItem

No	Method	Description
1	PackageId	Id of package
2	PackageName	Name of package

3	Price	Price of package
4	Discount	Discount of package
5	PriceAfterDiscount	Price of item after discount

3.2.2.24 TransactionCoins

No	Method	Description
1	Id	Id of transaction
2	UserId	ID of the user performing the transaction
3	AuthorId	Related author ID (if any)
4	PackageId	ID of the purchased package
5	PackageName	Name of the purchased package
6	Amount	Actual amount of coins in the transaction
7	OriginalPrice	Original price of the package
8	Description	Description of the transaction
9	CreatedAt	Timestamp of the transaction
10	OrderId	ID of the related order
11	Type	Type of transaction (enum TransactionCoinsType)

3.2.3 Sequence Diagram: Buy Game Product & Game Asset By Coins

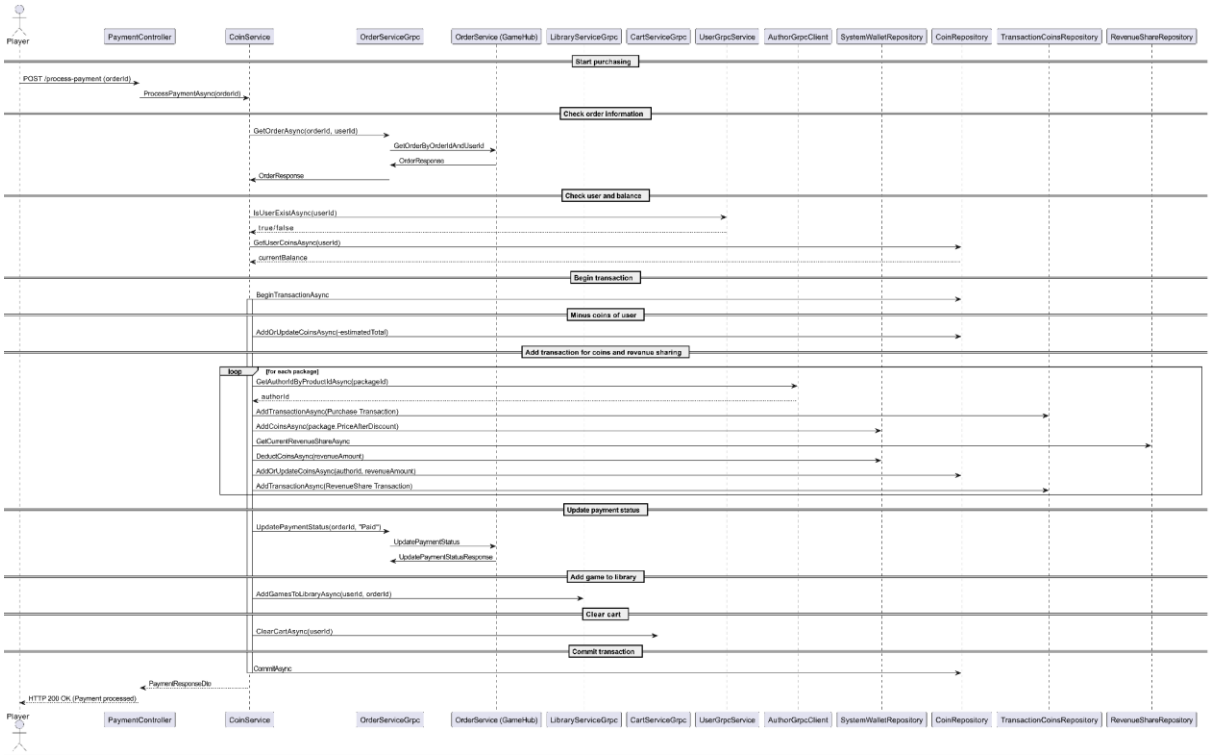

Figure 119 - Buy Game Product & Game Asset By Coins Sequence Diagram

[Payment sequence diagram]

3.3 Download Game/Asset Content

3.3.1 Class Diagram

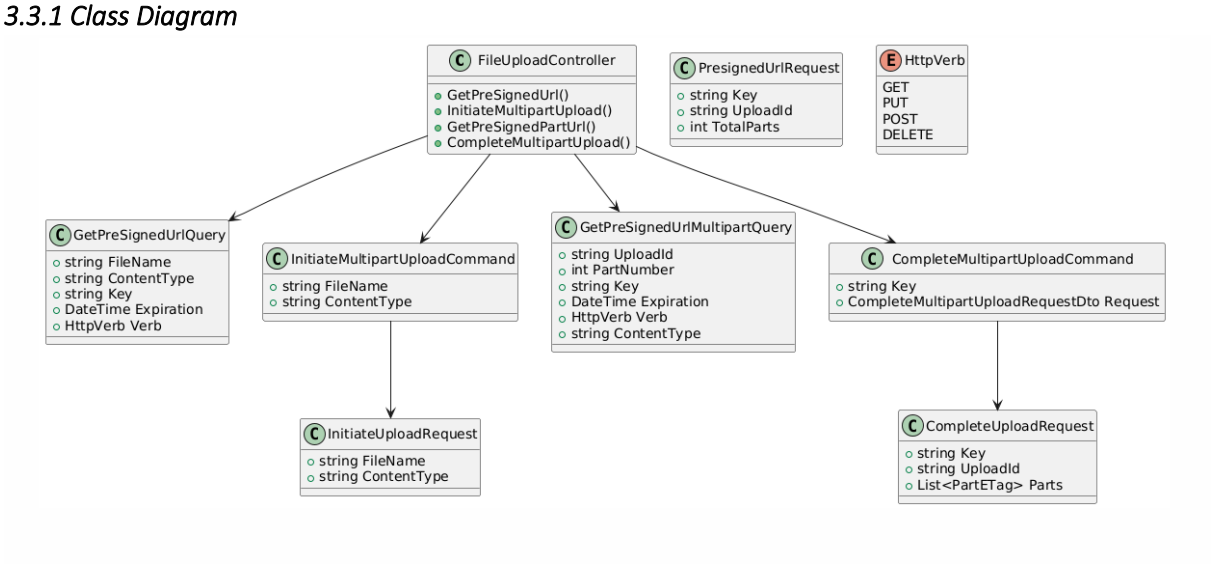

Figure 120 - Download Game/Asset Content Sequence Diagram

3.3.2 Class Diagram Specification

3.3.2.1. FileUploadController

No	Method	Description
1	GetPreSignedUrl()	Generates a presigned URL for file operations
2	InitiateMultipartUpload()	Initiates a multipart upload process

3	GetPreSignedPartUrl()	Generates presigned URL for a specific part of multipart upload
4	CompleteMultipartUpload()	Completes the multipart upload process

3.3.2.2. InitiateUploadRequest

Property	Type	Description
FileName	string	Name of the file to upload
ContentType	string	MIME type of the file

3.3.2.3. PresignedUrlRequest

Property	Type	Description
Key	string	Unique identifier for the file
UploadId	string	ID of the multipart upload
TotalParts	int	Total number of parts in the upload

3.3.2.4. CompleteUploadRequest

Property	Type	Description
Key	string	Unique identifier for the file
UploadId	string	ID of the multipart upload
Parts	List<PartETag>	List of uploaded parts with their ETags

3.3.2.5. GetPreSignedUrlQuery

Property	Type	Description
FileName	string	Name of the file
ContentType	string	MIME type of the file
Key	string	Unique identifier for the file
Expiration	DateTime	URL expiration time
Verb	HttpVerb	HTTP method for the URL

3.3.2.6. InitiateMultipartUploadCommand

Property	Type	Description
FileName	string	Name of the file
ContentType	string	MIME type of the file

3.3.2.7. GetPreSignedUrlMultipartQuery

Property	Type	Description
UploadId	string	ID of the multipart upload
PartNumber	int	Number of the part
Key	string	Unique identifier for the file
Expiration	DateTime	URL expiration time
Verb	HttpVerb	HTTP method for the URL
ContentType	string	MIME type of the file

3.3.2.8. CompleteMultipartUploadCommand

Property	Type	Description
Key	string	Unique identifier for the file
Request	CompleteMultipartUploadRequestDto	Request containing parts information

3.3.2.9. HttpVerb (Enum)

Value	Description

GET	HTTP GET method
PUT	HTTP PUT method
POST	HTTP POST method
DELETE	HTTP DELETE method

3.3.3 Sequence Diagram: Download Executable Source Game Build

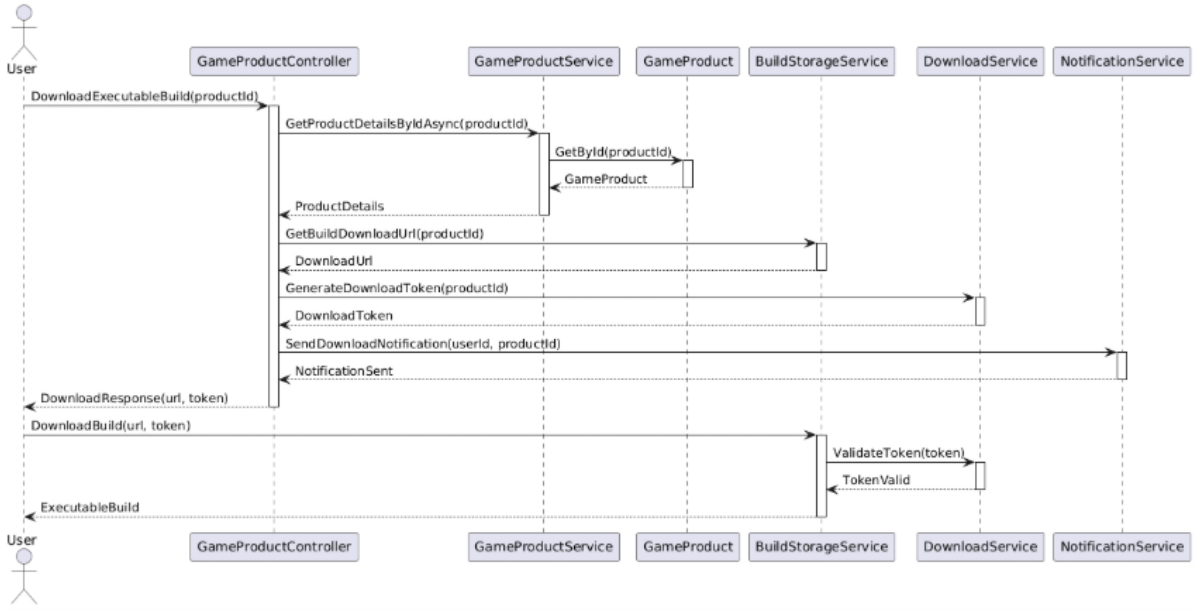

Figure 121 - Download Executable Source Game Build Sequence Diagram

3.3.4 Sequence Diagram: Download Asset Package File

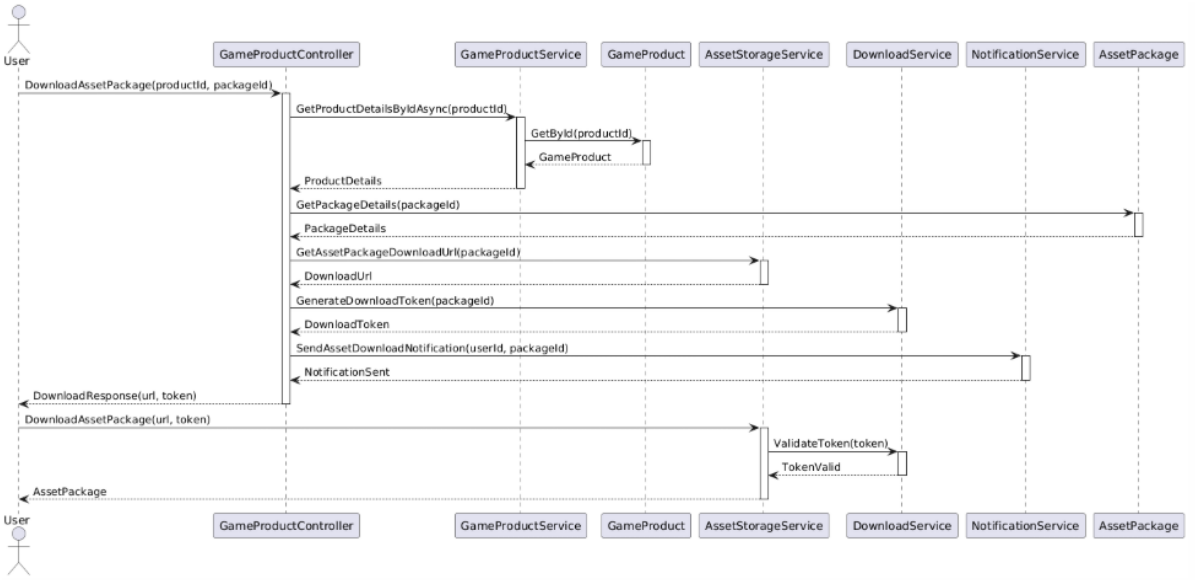

Figure 122 - Download Asset Package File Sequence Diagram

3.4 Sharing Revenue Share

3.4.1 Class Diagram

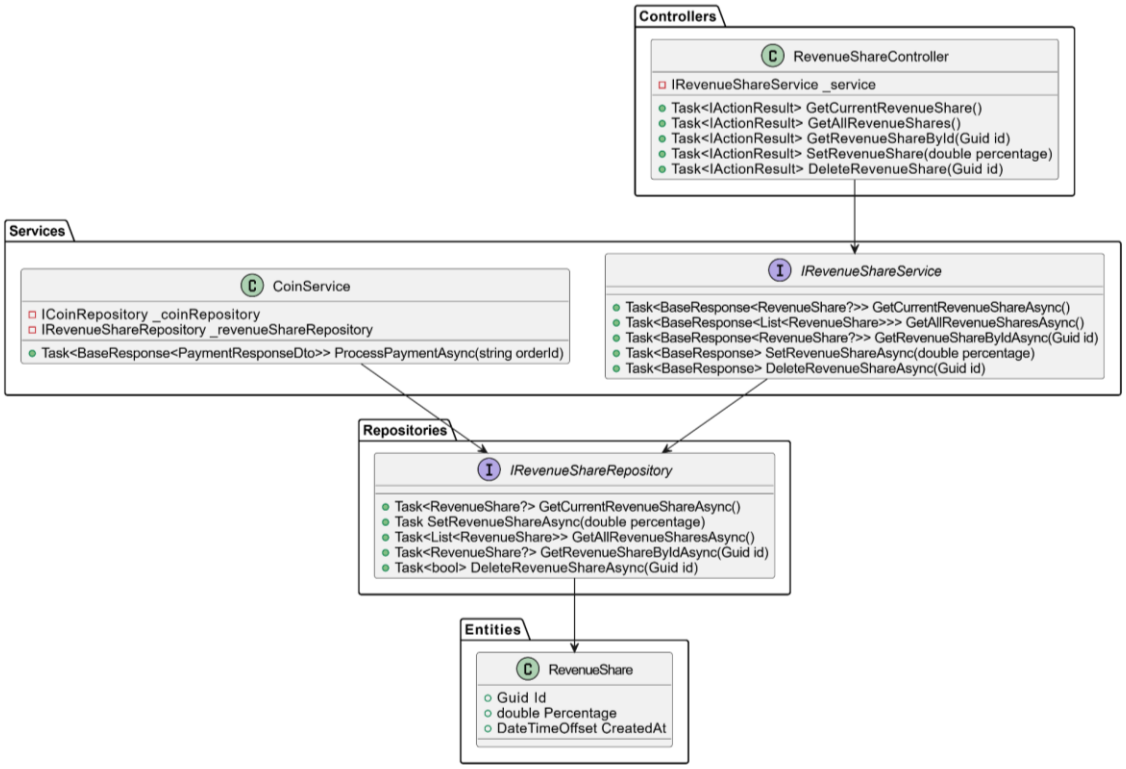

Figure 123 - Sharing Revenue Share

3.4.2 Class Diagram Specification

3.4.2.1 RevenueShareController

No	Method	Description
1	GetCurrentRevenueShare()	Handles HTTP GET to retrieve the most recent revenue share.
2	GetAllRevenueShares()	Handles HTTP GET to retrieve all revenue share entries.
3	GetRevenueShareById(id)	Handles HTTP GET to retrieve a specific revenue share by ID.
4	SetRevenueShare(percentage)	Handles HTTP POST to set a new revenue share percentage.
5	DeleteRevenueShare(id)	Handles HTTP DELETE to remove a revenue share entry by ID.

3.4.3 Sequence Diagram: Sharing Revenue Share From Purchased Game Product & Game Asset

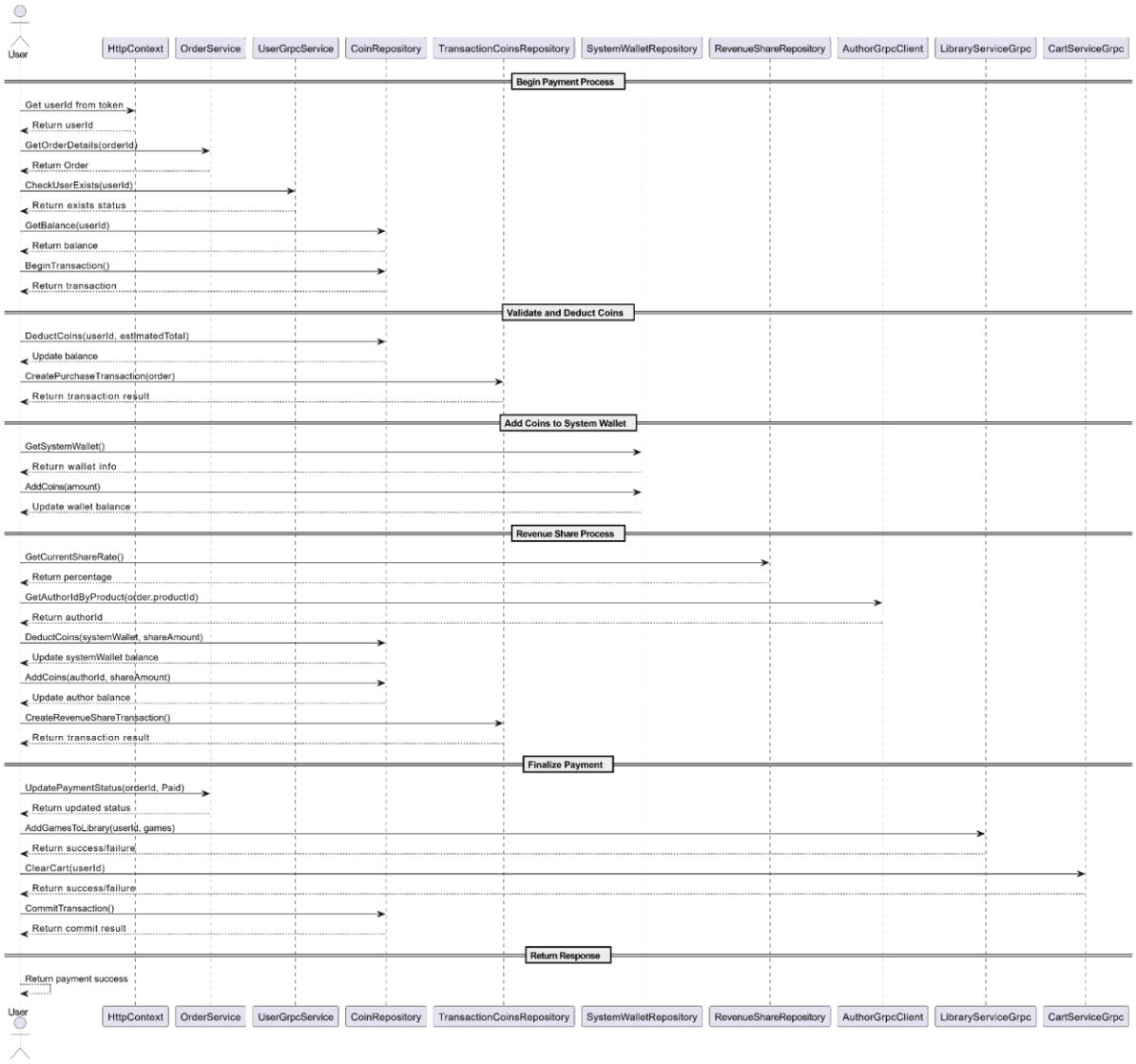

Figure 124 - Sharing Revenue Share From Purchased Game Product & Game Asset Sequence Diagram

3.5 Withdraw Coins

3.5.1 Class Diagram

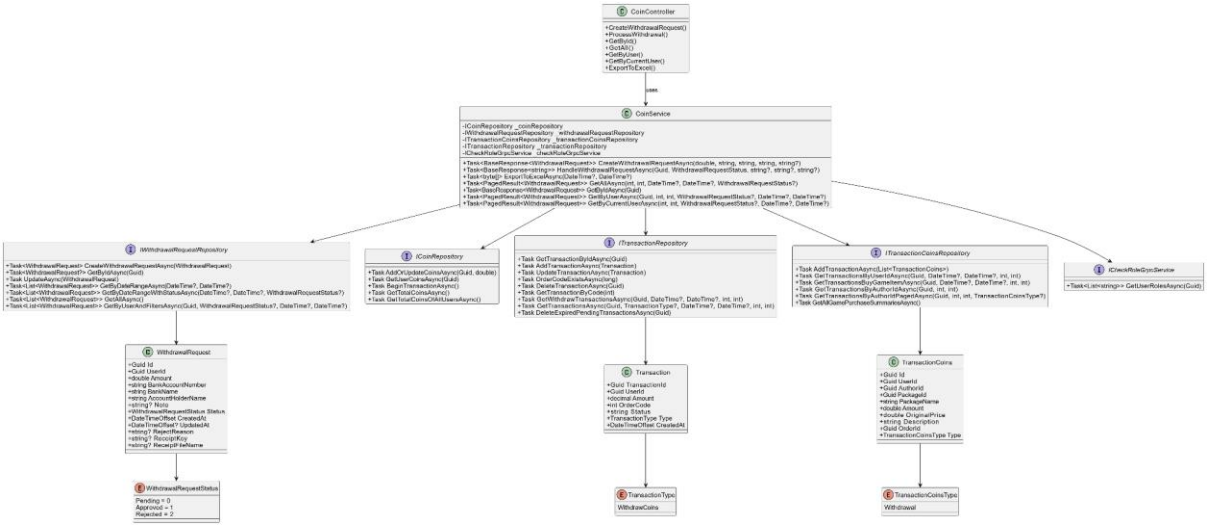

Figure 125 - Withdraw Coins Class Diagram

3.5.2 Class Diagram Specification

3.5.2.1 WithdrawController

No	Method	Description
1	CreateWithdrawalRequest()	Handles HTTP POST to create a new withdrawal request.
2	ProcessWithdrawal()	Handles HTTP PUT to approve or reject a withdrawal request, including uploading receipt.
3	GetById()	Handles HTTP GET to retrieve a specific withdrawal request by ID.
4	GetAll()	Handles HTTP GET to retrieve all withdrawal requests with optional filters (date, status).
5	GetByUser()	Handles HTTP GET to retrieve withdrawal requests by user ID and filters (status, date).
6	GetByCurrentUser()	Handles HTTP GET to retrieve withdrawal requests of the currently authenticated user.
7	ExportToExcel()	Handles HTTP GET to export withdrawal requests to an Excel file by date range.

3.5.2.2 WithdrawService

No	Method	Description
1	CreateWithdrawalRequestAsync(...)	Creates a new withdrawal request and logs the pending coin transaction.
2	HandleWithdrawalRequestAsync(...)	Approves or rejects a withdrawal request, updates status, and creates coin transactions.
3	ExportToExcelAsync(...)	Exports filtered withdrawal requests to an Excel file.
4	GetAllAsync(...)	Retrieves all withdrawal requests with pagination and filtering.
5	GetByIdAsync(id)	Retrieves a specific withdrawal request by ID.
6	GetByUserAsync(...)	Retrieves withdrawal requests by user ID with filters and pagination.
7	GetByCurrentUserAsync(...)	Retrieves withdrawal requests for the currently authenticated user.

3.5.2.3 WithdrawalRequestRepository

No	Method	Description
1	CreateWithdrawalRequestAsync(...)	Inserts a new withdrawal request into the database.
2	GetByIdAsync(id)	Retrieves a withdrawal request by ID.
3	UpdateAsync(...)	Updates a withdrawal request entity.
4	GetByDateRangeAsync(...)	Retrieves withdrawal requests within a date range.
5	GetByDateRangeWithStatusAsync(...)	Retrieves withdrawal requests by date and status filters.
6	GetAllAsync()	Retrieves all withdrawal requests.
7	GetByUserAndFiltersAsync(...)	Retrieves withdrawal requests by user ID with status and date filters.

3.5.2.4 WithdrawalRequest Entity

No	Property	Type	Description
1	Id	Guid	Unique identifier of the withdrawal request.
2	UserId	Guid	ID of the user making the request.
3	Amount	double	Amount of coins requested to

			withdraw.
4	BankAccountNumber	string	Bank account number for the ltransfer.
5	BankName	string	Name of the bank.
6	AccountHolderName	string	Name of the account holder.
7	Note	string?	Optional user note.
8	Status	WithdrawalRequestStatus	Current status (Pending, Approved, Rejected).
9	CreatedAt	DateTimeOffset	Creation timestamp.
10	UpdatedAt	DateTimeOffset?	Last update timestamp (if any).
11	RejectReason	string?	Reason for rejection.
12	ReceiptKey	string?	Cloud storage key for the receipt file.
13	ReceiptFileName	string?	Original name of the uploaded receipt file.

3.5.2 Sequence Diagram: Send Request Withdraw

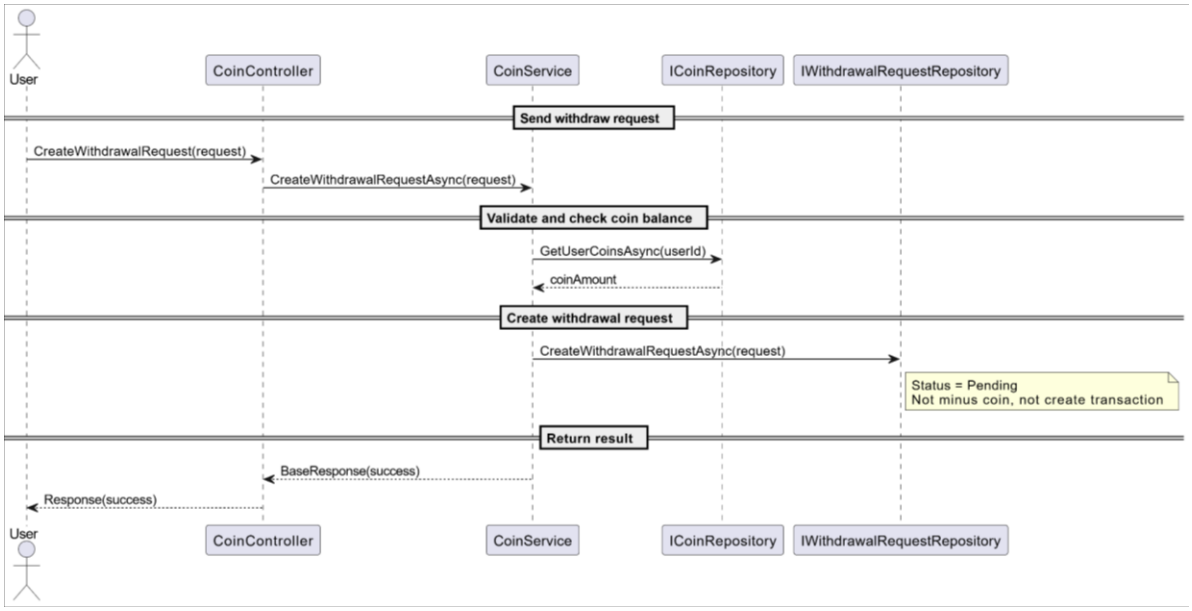

Figure 126 - Send Request Withdraw Sequence Diagram

3.5.3 Sequence Diagram: Export Withdraw Request To File

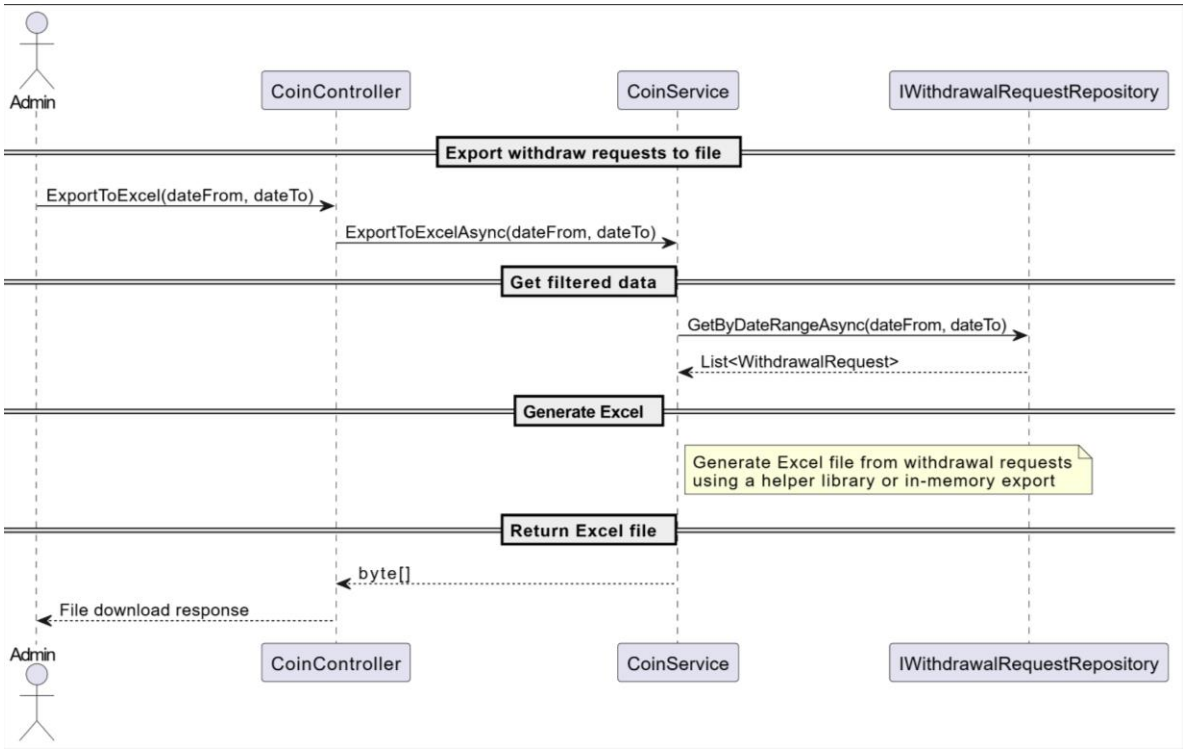

Figure 127 - Export Withdraw Request To File

3.5.4 Sequence Diagram: Approve / Reject Withdraw Request

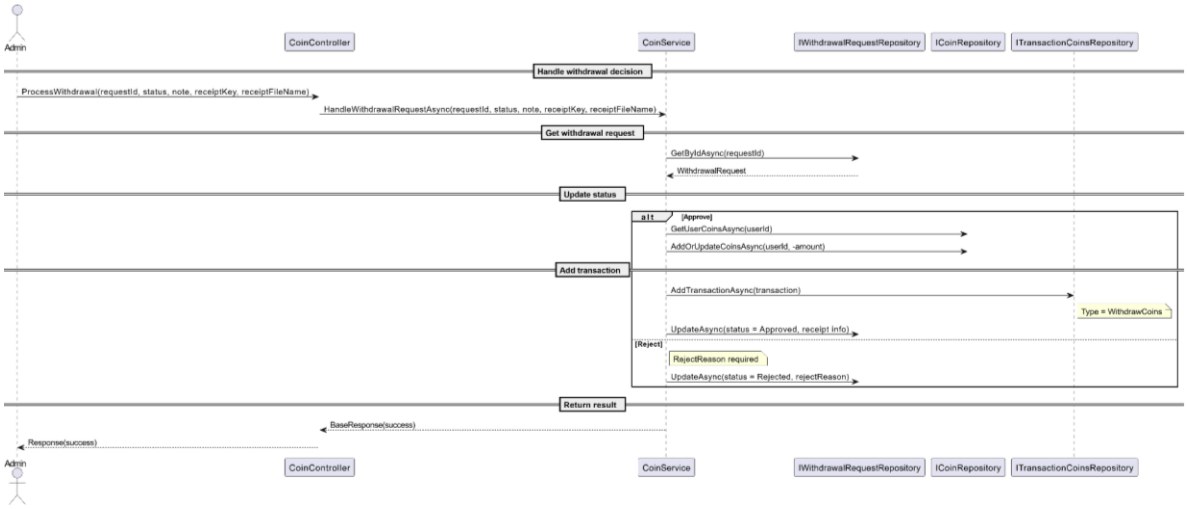

Figure 128 - Approve / Reject Withdraw Request

3.6 Search, Sort, Filter On Game Product

3.6.1 Class Diagram

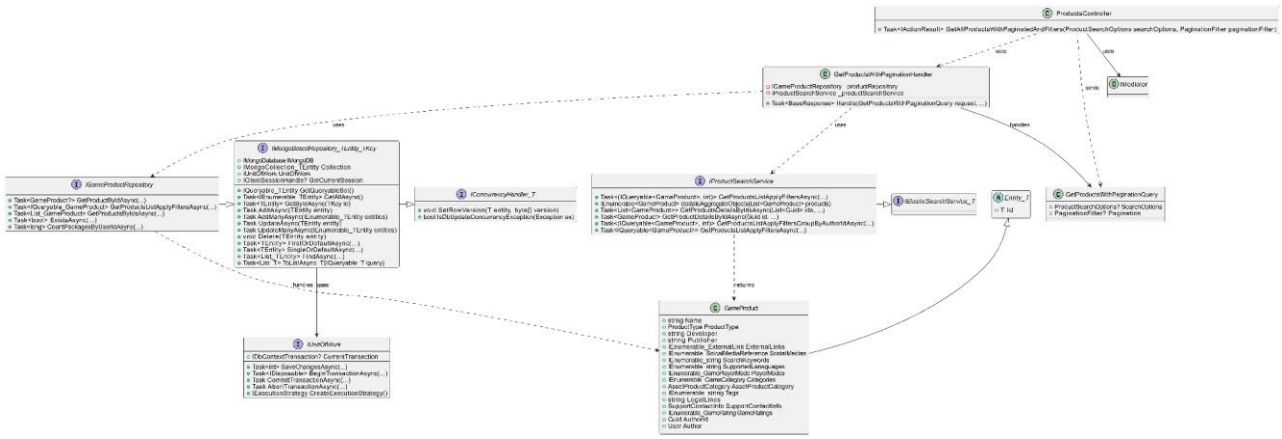

Figure 129 - Search, Sort, Filter On Game Product Class Diagram

Search, Sort, Filter On Game Product

3.6.2 Class Diagram Specification

3.6.2.1 GameProductController

No	Method	Description
1	GetAllProductsWithPaginatedAndFilters(...)	Handles HTTP GET to retrieve paginated list of GameProducts with filters/search/sort
2	GetProductsListApplyFiltersAsync(...)	Queries MongoDB with filters like ProductType, Languages, Categories, Tags
3	GetProductsListApplyFiltersAsync(...)	Applies ElasticSearch-based search + sort on fields like Name, Developer, Tags
4	GetProductsListApplyFiltersGroupByAuthorIdAsync(...)	Returns products grouped by AuthorId, with optional filters
5	GetProductsDetailsByIdsAsync(...)	Retrieves full GameProduct details by list of IDs
6	GetProductDetailsByIdAsync(Guid id, ...)	Retrieves detailed GameProduct by ID
7	UpdateAggregateObjects(List<GameProduct> products)	Enhances product list with derived/searchable fields from ElasticSearch

3.6.2.2 ProductSearchService Interface

No	Method	Description
1	GetProductsListApplyFiltersAsync(...)	Queries ElasticSearch with filters

2	UpdateAggregateObjects(List<GameProduct> products)	Enhances searchability of product list
3	GetProductsDetailsByIdsAsync(List<Guid> ids, ...)	Retrieves detailed GameProducts by list of IDs
4	GetProductDetailsByIdAsync(Guid id, ...)	Retrieves single product detail from ElasticSearch
5	GetProductsListApplyFiltersGroupByAuthorIdAsync(...)	Returns product list grouped by AuthorId

3.6.2.3 GameProductRepository

No	Method	Description
1	GetProductByIdAsync(Guid id)	Gets a GameProduct by ID
2	GetProductsListApplyFiltersAsync(...)	Retrieves GameProducts filtered via Mongo
3	GetProductsByIdsAsync(List<Guid> ids)	Retrieves list of GameProducts by list of IDs
4	ExistsAsync(Expression<Func<GameProduct, bool>> filter)	Checks if any product matches a filter
5	CountPackagesByUserIdAsync(Guid userId)	Count how many products belong to a specific user

3.6.2.4 GameProduct

No	Property	Type	Description
1	Id	Guid	Unique identifier of the GameProduct
2	Name	string	Name of the game
3	ProductType	ProductType	Enum indicating product type
4	Developer	string	Developer name
5	Publisher	string	Publisher name
6	ExternalLinks	IEnumerable<ExternalLink>	External reference links
7	SocialMedias	IEnumerable<SocialMediaReference>	Social media references
8	SearchKeywords	IEnumerable<string>	Searchable keywords
9	SupportedLanguages	IEnumerable<string>	Languages supported

10	PlayerModes	IEnumerable<GamePlayerMode>	Game player modes
11	Categories	IEnumerable<GameCategory>	Game categories
12	AssetProductCategory	AssetProductCategory	Linked asset product info
13	Tags	IEnumerable<string>	Searchable tags
14	LegalLines	string	Legal information
15	SupportContactInfo	SupportContactInfo	Contact info for support
16	GameRatings	IEnumerable<GameRating>	Rating details
17	AuthorId	Guid	Id of the creator/author
18	Author	User	Reference to the author

3.6.3 Sequence Diagram: Search, Sort, Filter On Game Product

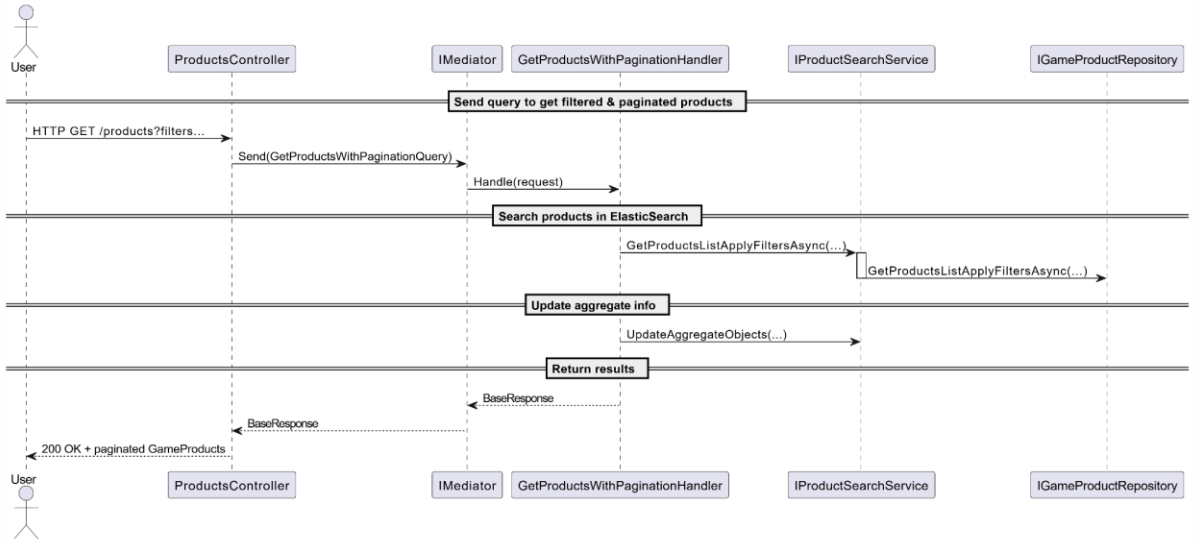

Figure 130 - Search, Sort, Filter On Game Product Sequence Diagram

3.7 Send Notification

3.7.1 Class Diagram

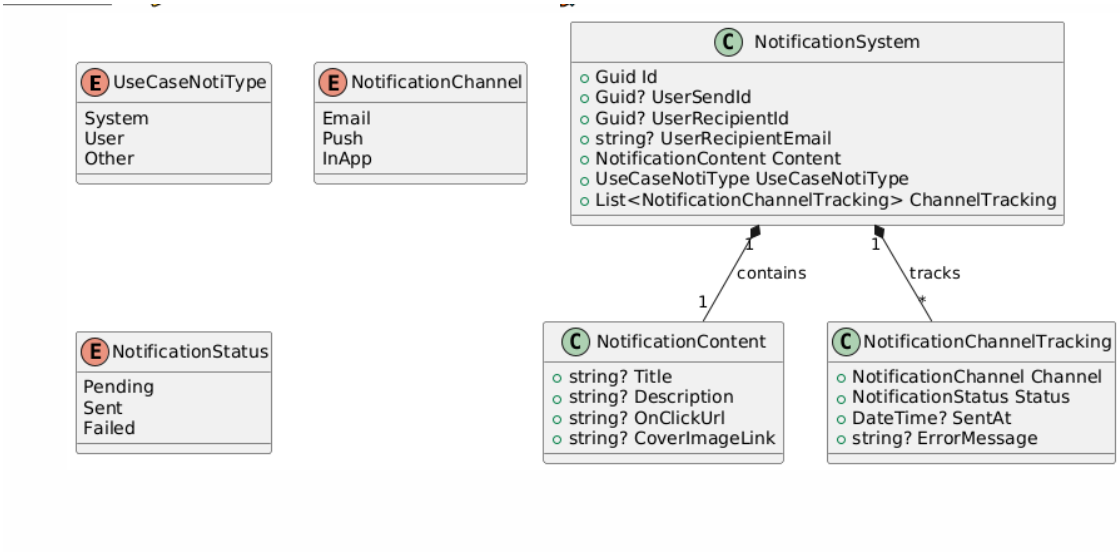

Figure 131 – Send Notification Class Diagram

3.7.2 Class Diagram Specification

3.7.2.1 NotificationSystem

No	Method	Description
1	CreateNotification(...)	Creates a new notification with sender, recipient, and content details
2	AddChannelTracking(...)	Adds a new channel tracking entry for the notification
3	UpdateChannelStatus(...)	Updates the delivery status for a specific channel
4	GetNotificationStatus()	Retrieves the overall status of the notification across all channels
5	GetFailedChannels()	Returns list of channels where notification delivery failed
6	GetPendingChannels()	Returns list of channels where notification is still pending
7	GetSuccessfullyDeliveredChannels()	Returns list of channels where notification was successfully delivered

3.7.2.2 NotificationContent

No	Method	Description
1	CreateContent(...)	Creates notification content with title, description, and optional URL/imagedetails
2	ValidateContent()	Validates that required content fields are present
3	GetFormattedContent()	Validates that required content fields are present
4	GetContentPreview()	Returns a preview of the notification content

3.7.2.3 NotificationChannelTracking

No	Method	Description
1	CreateTracking(...)	Creates a new tracking entry for a specific channel
2	Creates a new tracking entry for a specific channel	Creates a new tracking entry for a specific channel
3	UpdateStatus(...)	Updates the delivery status and timestamp
4	RecordError(...)	Records error details when delivery fails
5	GetDeliveryMetrics()	Returns delivery metrics for the channel
6	IsRetryable()	Determines if the failed delivery can be retried

3.7.2.4 UseCaseNotiType (Enum)

No	Value	Description
1	System	System-generated notifications
2	User	User-generated notifications
3	Other	Other types of notifications

3.7.2.5 NotificationChannel (Enum)

No	Value	Description
1	Email	Email delivery channel
2	SMS	SMS delivery channel
3	Push	Push notification channel
4	InApp	In-application notification channel

3.7.2.6 NotificationStatus (Enum)

No	Value	Description
1	Pending	Notification is queued for delivery |
2	Sent	Notification was successfully delivered
3	Failed	Notification delivery failed

3.7.3 Sequence Diagram: Send Notification

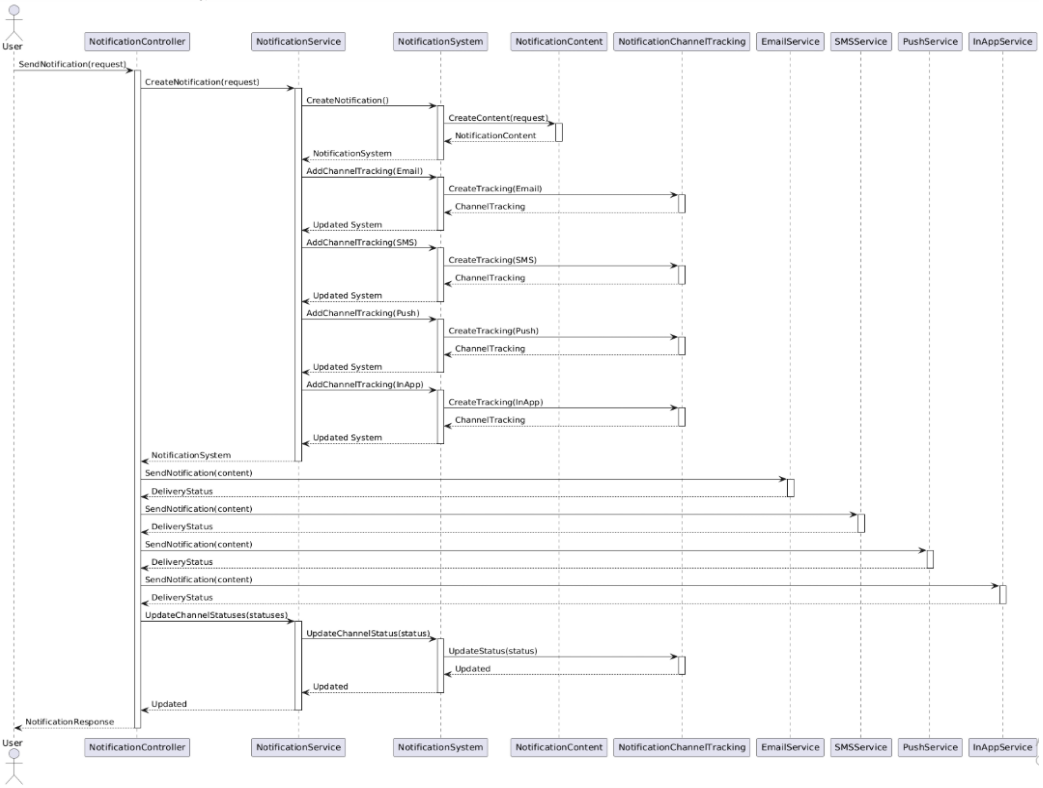

Figure 132 – Send Notification Sequence Diagram

V. Software Testing Documentation

1. Scope of Testing

1.1 In Scope

Function	Role(s) of Use	Description
User Registration & Login	Guest; Customer; Developer; Designer; Admin	Sign-up new accounts; log in and log out; reset or change passwords; view and update own profile.
FPT Student Verification	Guest; Moderator; System Handler	Scan student card, request and approve/reject FPT-student status; then register as Developer/Designer.
Notifications	System; Admin; Developer; Moderator; All Roles	Send peer-to-peer, broadcast, email, and OTP notifications; view notification history.
System Accounts & Permissions	Admin; System Handler	Create, view, update, delete, ban/unban system accounts; create and manage roles and permissions.
Platform Configuration	Admin	Set global preferences and platform-wide settings.
API Documentation	Moderator	Create, update, delete, publish and view API documentation.
Project & Player Account Integration	Developer	Create, view, update, delete projects; integrate external game-player accounts and in-game items.
Game Product Management	Developer; Moderator; Customer	Create, update, delete products/builds; schedule or immediate release; download builds.
Game Asset Management	Designer; Moderator; Customer	Create, update, delete assets and files; schedule or immediate release; download asset files.
Release Workflow	Developer; Designer; Moderator	Submit release requests; approve/reject; track status; ban/unban released items.
Discount Campaigns	Moderator; Developer; Designer; System Handler	Create, edit, publish, schedule campaigns; apply or cancel campaigns on products/assets.
Storefront Browsing & Purchasing	Customer	Search, sort, filter catalog; view details; manage cart and wishlist; make purchases.
Coin & Wallet Management	Customer; Developer; Designer; Admin	Purchase coins; view account/platform balances; track coin-coin and coin-money transactions; monitor sales.
Library, Feedback	Customer; Moderator	View/manage purchased library; submit,

& Reporting		update or delete reviews; view all reports on items.
Revenue & Withdrawals	Admin; Developer; Designer	Configure revenue sharing; export revenue data; submit, cancel, track and approve withdrawal requests.
Subscription Management	Admin; Developer; Designer	Create, update, remove subscription plans; view and purchase subscriptions.
News & Announcements	Moderator	Create, update, delete, publish and list platform news posts.
Support Tickets	Moderator; Customer; Developer; Designer	Browse and view ticket list and details; moderator resolves support issues.

Table 72 – In Scope Testing

1.2 Out-Of-Scope

These items will not be covered by the current testing engagement:

- Integration with third-party payment provider's internal systems (beyond basic API smoke-tests)
- End-to-end performance/load testing past defined service-level thresholds
- Security penetration testing (to be handled by dedicated security audit)
- Support for legacy browsers (e.g. Internet Explorer 11)
- Offline capabilities or mobile features outside of authenticated flows
- Accessibility compliance testing (WCAG, etc.)

1.3 Constraints and Assumptions

- Resource Availability: Not all planned test scenarios may be executed if key personnel become unavailable.
- Environment Stability: Unplanned outages or crashes of test servers may delay test execution.
- Requirements Volatility: Changes in requirements or design late in the cycle may require rework of test cases and schedules.
- Data Availability: Realistic test data for certain edge-cases (e.g. large transaction volumes) may be limited.
- Tooling Limits: Automation coverage depends on availability and maturity of chosen frameworks

2. Test Strategy

2.1 Testing Types

- Functional Testing – Verify each feature behaves according to specification.
- Integration Testing – Ensure modules and external APIs interoperate correctly.
- System Testing – Validate end-to-end business flows on web and mobile.
- Regression Testing – Re-run existing suites after each build to catch unintended breaks.
- Performance Testing – Measure responsiveness under normal and peak loads (smoke level).
- Usability / UI Testing – Confirm user interfaces are intuitive and consistent.
- Security / Access Control Testing – Verify authentication, authorization, and data privacy controls.

2.2 Test Levels

1. Unit Tests (Developer-owned)
2. Integration Tests (Service-to-service and database interactions)
3. System Tests (End-to-end scenarios from UI/UX perspective)
4. Acceptance Tests (Business stakeholder validation against requirements)

2.3 Supporting Tools

- Test Management: Jira / Zephyr for planning and tracking test execution
- Defect Tracking: Jira for logging and triaging bugs
- Automation Frameworks: Selenium (web)
- Performance Monitoring: JMeter or Gatling (smoke-level scripts)
- CI/CD Integration: GitHub Actions / Jenkins for running automated suites on each push

3. Test Plan

3.1 Human Resources

Role	Responsibilities
Leader	Overall test coordination, progress reporting
Members	Support unit/integration testing and defect resolution

Table 73 – Human Resources

3.2 Test Environment

- Web App: Windows 10 / macOS Safari, Chrome, Firefox
- Back-end Services: Deployed to test cluster (API, DB, message brokers)

3.3 Test Milestones

Milestone	Target Date	Deliverable
Test Strategy Sign-Off	Week 9	Strategy & plan document approved
Test Case Development	Week 10-11	All high-priority test cases authored
Environment Readiness	Week 11	Test environments fully configured
Functional Test Execution	Week 11–12	Execution report for all core flows
Automation Suite Completion	Week 12	Automated smoke/regression ready
Performance Smoke Testing	Week 13	Performance baseline report
User Acceptance Testing (UAT)	Week 14	UAT sign-off from business owners

Table 74 – Test Milestones

4. Test Cases

5. Test Reports

No	Function Name	Sheet Name	Description	Pre-Condition
1	Function Request Creation	Feature 1		
2	Function Moderator Review	Feature 2		
3	Function Account Creation	Feature 3		
4	Function Purchase Game Products and Game Assets	Feature 4		
5	Function Download Game Contents from Library	Feature 5		
6	Function Create Discount	Feature 6		
7	Function Apply Discount	Feature 7		
8	Function Integrate APIs	Feature 8		
9	Function create request withdraw	Feature 9		
10	Function Admin approval	Feature 10		
11	Function Developer Create Game Product	Feature 11		
12	Function Developer Create Game package	Feature 12		
13	Function Designer Create Asset	Feature 13		
14	Function Designer Create Asset Packages	Feature 14		

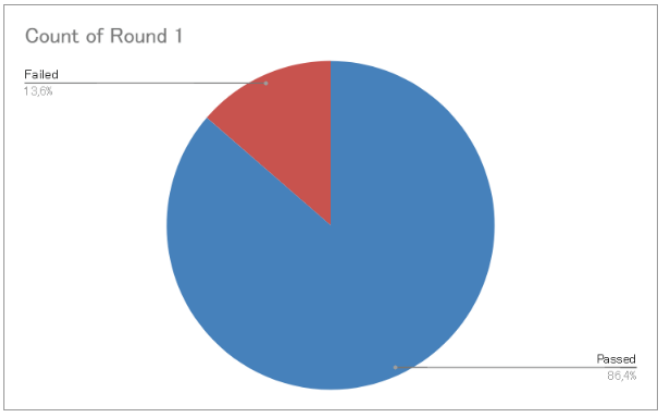

Figure 133 – Test Reports

VI. Release Package & User Guides

1. Deliverable Package

No.	Deliverable Item	Description
1	Source Codes	Source Code Base Front End Project, Backend Project
2	Database Script(s)	Database Script Of MySQL Database
3	Final Report Document	Final Report Document
4	Test Cases Document	Test Case Report
5	Slide	Slide Show On Capstone Project

Table 75 - Deliverable Package

2. Installation Guides

2.1 System Requirements

2.1.1 Web Application

As we host the Front-End layer on Amazon Web Service with Amplify (an AWS-managed infrastructure service to deploy web applications) so we don’t need to care about hardware requirements of web application.

No.	Package name	Minimum version
1	Node	>= Node.js 18.x
2	npm	>= 10.9.2
3	Next.js	14.2.1
4	React & ReactDOM	react 18, react-dom 18
5	tailwindcss	3.4.1
6	typescript	5.8.2
7	apexcharts	4.5.0
8	babel/core	7.26.10
9	eslint	8.57.0
10	prettier	3.4.2
11	sentry/nextjs	9.12.0
12	.Net SDK	>= 7.0

Table 76 - Web Application Requirements

2.1.2 Backend Services

No.	Component	Minimum configuration
1	vCPU	2
2	Memory	8 GB
3	Disk	20 GB
4	Kubernetes version	>= v1.31.5
5	Container runtime	containerd://1.7.25
6	Kernel Version	6.1.129-138.220.amzn2023.x86_64

Table 77 – Backend Requirements

2.1.3 Database

No.	Component	Minimum configuration
1	vCPU	2
2	Memory	2 GB
3	Disk	20 GB
4	Number of instance	3
5	Engine version	>= MySQL 8.0.40

Table 78 – Database Requirements

2.2 Installation Instruction

2.2.1 Setup infrastructure

- Create a private network (VPC) for production environments.
- Create 3 database instances (RDS) for the database layer in the private subnet of VPC.
- Create 3 virtual machines (EC2) to use as proxy servers that allow developers access private databases from local machines.
- Set-up a kubernetes cluster (EKS) to run backend services.
- Connect github with CI/CD service (Amplify) to deploy the frontend layer.
- Register domain, request certify authority certification for domains, configure content delivery network (CDN) and custom domain point to frontend, backend layer.
- Config storage service (S3) to store website assets.
- Config mail service (SES).
- Config clickstream database (DynamoDB).
- Create a MongoDB cluster.
- Config terraform to provision modules of feature services.

2.2.2 Web Application

- Edit .env file.
- Install necessary packages: npm install.
- Next.js Build Command: npm run build.
- Development Command: npm run dev.
- Start Command: npm run start.

2.2.3 Backend Service

- Edit .env file and appsettings.json.
- Start Command: docker compose up – build