CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

CEBU INSTITUTE OF TECHNOLOGY UNIVERSITY

COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for

Darknet Duel

Change History

Version	Date	Description	Author
1.0	14 March, 2025	Initial Document	Entirety of Team 27 of Capstone 1
2.0	14 June, 2025	Shift from Spring Boot as Backend to Express.js with boardgame.io. Reworked all modules.	Entirety of Team 27 of Capstone 1
3.0	20 September, 2025	Apply requirements refactor: Add lore video, interactive tutorial, and various additions	Entirety of Team 27 of Capstone 1

Table of Contents

1. Introduction

1.1. Purpose

This Software Requirements Specification (SRS) describes the requirements for the Darknet Duel web-based card game. The document is intended to guide the development team, quality assurance testers, project stakeholders, and game designers in understanding the goals, features, and constraints of the system. The SRS aims to provide a clear and comprehensive description of the system's intended functionality, architecture, and user experience, ensuring that all parties share a common understanding of the project's direction and deliverables.

1.2. Scope

Darknet Duel is a web-based, real-time, turn-based card game with the following features:

- Core game engine (boardgame.io) for card mechanics and turn logic
- REST API backend for authentication, user management, persistence, and admin tools
- React-based frontend for all user interactions
- Lobby and matchmaking system
- Real-time multiplayer gameplay via WebSocket
- Game state management and result tracking
- In-game currency, store, and profile management
- Admin and moderator dashboards

1.3. Definitions, Acronyms and Abbreviations

- AP: Action Points
- JWT: JSON Web Token
- REST: Representational State Transfer
- API: Application Programming Interface
- WebSocket: Real-time communication protocol
- Red Team: Attacker role
- Blue Team: Defender role
- Infrastructure: Cards representing assets
- ELO: Rating system for player skill
- Creds/Crypts: In-game currencies

1.4. References

The following documents and resources are referenced in this SRS:

- boardgame.io Documentation (boardgame.io Project, https://boardgame.io/documentation)
- React.js Documentation (Meta Platforms, Inc., https://react.dev)
- Express.js Documentation (OpenJS Foundation, https://expressjs.com/en/5x/api.html)
- MySQL Documentation (Oracle Corporation, https://dev.mysql.com/doc/)
- Socket.IO Documentation (Socket.IO Project, https://socket.io/docs/)
- Swagger/OpenAPI 3.0 (OpenAPI Initiative, https://swagger.io/specification/)
- OWASP ZAP (Zed Attack Proxy) (OASP Foundation, https://www.zaproxy.org/)
- OWASP ASVS (Application Security Verification Standard) (OWASP Foundation, https://owasp.org/www-project-application-security-verification-standard/)

2. Overall Description

2.1. Product perspective

The system is composed of three main parts:

1. Backend Server (Node.js/Express/MySQL)
    - User authentication (JWT)
    - User management (profile, stats, currency)
    - Game result and history persistence
    - Store, purchases, and decoration management
    - Admin/moderator tools
    - REST API and WebSocket endpoints
    - API documentation with Swagger/OpenAPI for 100% of REST endpoints
    - Rate limiting policies to prevent abuse
    - Audit logging for administrative actions
    - Security headers (Helmet) and configured CORS policies
    - Input validation/sanitization for all inbound data
    - GDPR features (data export and deletion)
    - Session timeout and automatic logout

2. Game Server (Node.js/boardgame.io)
    - Implements all game logic, state, and turn management
    - Handles real-time multiplayer via WebSockets
    - Synchronizes results with backend server
    - Manages lobbies, player actions, and game phases

3. Frontend (React/TypeScript)
    - User interface for all features (auth, lobby, game, store, admin)
    - Real-time updates via WebSockets
    - Responsive, cyberpunk-themed design
    - Role-based access (player, admin, moderator)

The system consists of the following major modules:

- Module 1: User Management and Authentication (Backend + Frontend)
    - Transaction 1.1: User registration (username, email, password, validation, password hashing)

    - Transaction 1.2: User login (JWT issuance, session management)
    - Transaction 1.3: Profile management (view/edit profile, avatar upload, stats)
    - Transaction 1.4: Role-based access (admin, moderator, player)

- Module 2: Lobby and Matchmaking (Game Server + Frontend)
    - Transaction 2.1: Lobby browser
    - Transaction 2.2: Create/join/leave lobbies
    - Transaction 2.3: Real-time lobby updates
    - Transaction 2.4: Lobby chat

- Module 3: Real-Time Multiplayer Game (Game Server + Frontend)
    - Transaction 3.1: Game creation and initialization
    - Transaction 3.2: Turn-based gameplay, AP allocation, card play, targeting
    - Transaction 3.3: Real-time state synchronization (boardgame.io, Socket.IO)
    - Transaction 3.4: Disconnection/reconnection handling
    - Transaction 3.5: Game state persistence and recovery

- Module 4: Card System and Game Logic (Game Server)
    - Transaction 4.1: Card play, targeting, and effect resolution
    - Transaction 4.2: Infrastructure state tracking (secure, vulnerable, compromised, fortified)
    - Transaction 4.3: Game rules enforcement (AP, hand size, win conditions)
    - Transaction 4.4: Game state visualization (frontend)

- Module 5: Game Statistics and Result Tracking (Backend + Frontend)
    - Transaction 5.1: Match result display (victory/defeat, win condition)
    - Transaction 5.2: Player performance statistics (ELO, win/loss, charts)
    - Transaction 5.3: Match history storage and browsing

- Module 6: Store and Currency (Backend + Frontend)
    - Transaction 6.1: In-game currency management (Creds, Crypts)
    - Transaction 6.2: Store browsing, item purchase, and application of decoration
    - Transaction 6.3: Payment integration (for top-up)

- Module 7: Admin and Moderation Tools (Backend + Frontend)
    - Transaction 7.1: User search, ban, and moderation
    - Transaction 7.2: Report management (submit, review, resolve)
    - Transaction 7.3: System logs and audit trails

- Transaction 7.4: User modification (Modify username, email, password)

- Module 8: First-Time Experience (Frontend)
  - Transaction 8.1: Lore video playback for first-time users
  - Transaction 8.2: Interactive tutorial flow following lore video completion

2.2. User characteristics

User Types:

- Registered Player:
  - Register/login, join/create lobbies, play games, manage profile, view history, purchase items
- Admin:
  - All player privileges, plus user management, banning, report handling, and log viewing
- Moderator:
  - All admin privileges, except user management

2.3. Constraints

1. Technical Constraints

- Web-based, responsive design (min 1280x720, primary device is Desktops and Laptops)
- Tech stack: Node.js, TypeScript, Express, MySQL, React, boardgame.io, Socket.IO
- JWT for authentication, REST for API, WebSocket for real-time communication
- Pre-defined card set (no custom/trading)
- Secure, scalable, and performant (see Non-Functional Requirements)

2. Game Design Constraints

- Maximum of 7 cards in hand
- 3 AP per turn for Defender (Blue Team), 2 AP per turn for Attacker (Red Team)
- 15 rounds maximum per game
- 5 infrastructure cards in play
- No custom card creation or trading

3. Performance Constraints

- Card action resolution < 1 second
- Game initialization < 3 seconds
- Maximum 2 seconds latency for real-time actions
- Support for 1000+ concurrent users
- Database queries < 500ms

2.4. Assumptions and dependencies

It is assumed that users have access to modern web browsers and stable internet connections. The system relies on the availability of web hosting services, a MySQL database server, and the Node.js runtime environment. The backend and game server must be able to communicate securely and reliably, and the system must be compatible with a range of devices and screen sizes. It is also assumed that users possess a basic understanding of card game mechanics and, ideally, some familiarity with cybersecurity concepts, although the game is designed to be accessible to newcomers.

3. Specific Requirements

3.1. External interface requirements

3.1.1. Hardware interfaces

- Desktop / Laptop
- Keyboard and mouse
- Audio output

3.1.2. Software interfaces

- Modern browsers (Chrome 100+, Firefox 100+, Safari 14+, Edge 100+)
- REST API (Express.js)
- WebSocket (Socket.IO)
- MySQL database
- JWT authentication

3.1.3. Communications interfaces

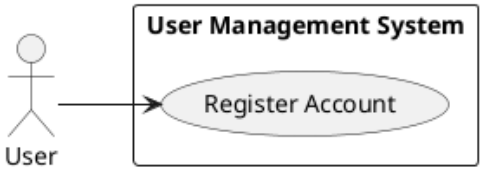
- HTTPS for all communications
- REST API for user management, store, history, etc.
- WebSocket for real-time game state and chat

3.2. Functional requirements

Module 1: User Management and Authentication (Backend + Frontend)

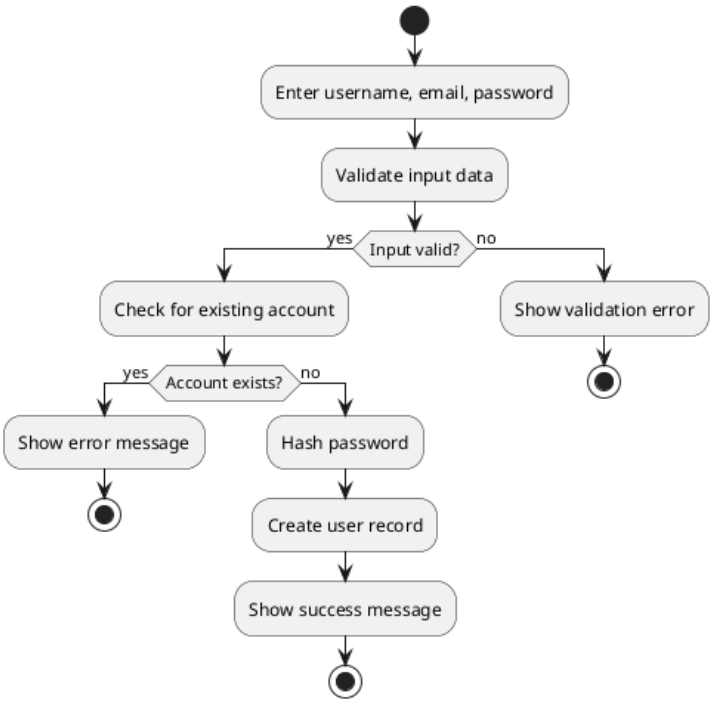
1.1 User registration

- *Use Case Diagram*



- *Use Case Description*
  Allows a new user to create an account by providing a username, email, and password. The system validates the input, checks for existing accounts, hashes the password, creates a new user record, and confirms successful registration.

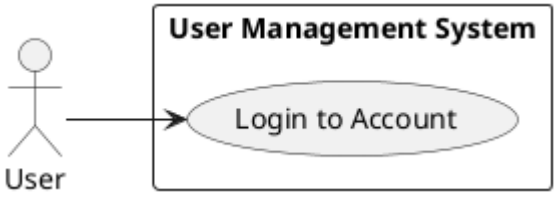
- *Activity Diagram*



- *Wireframe*

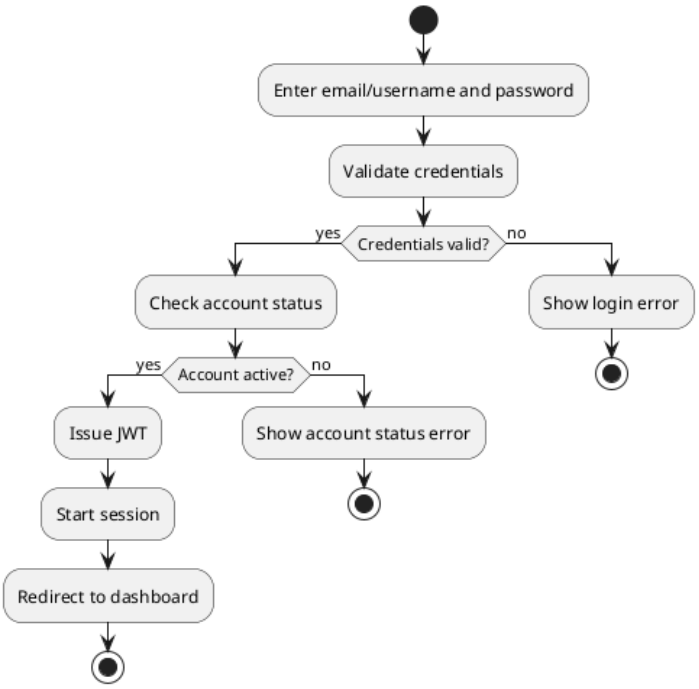
1.2 User login

- *Use Case Diagram*

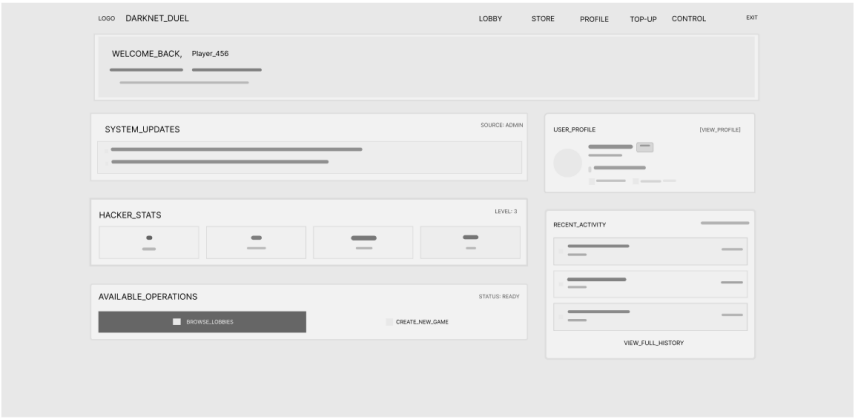


- *Use Case Description*
  Authenticates an existing user by verifying credentials and issuing a JWT for session management.

- *Activity Diagram*

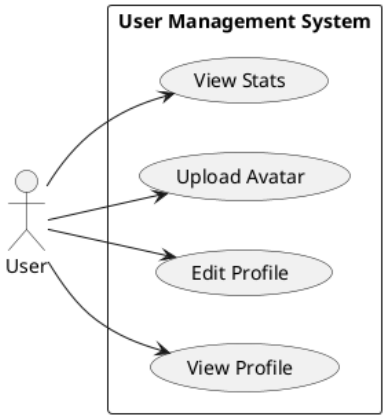


- *Wireframe*



1.3 Profile Management

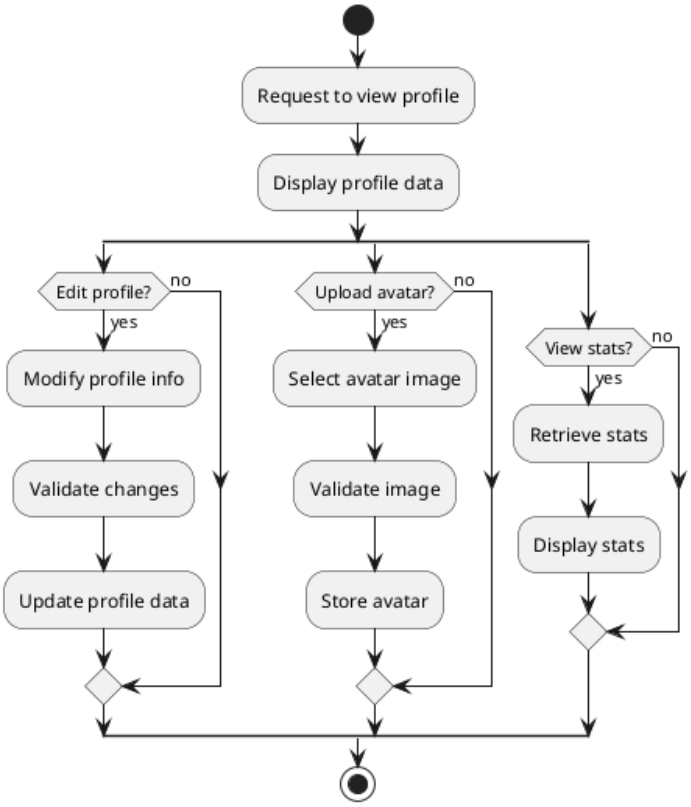
- *Use Case Diagram*



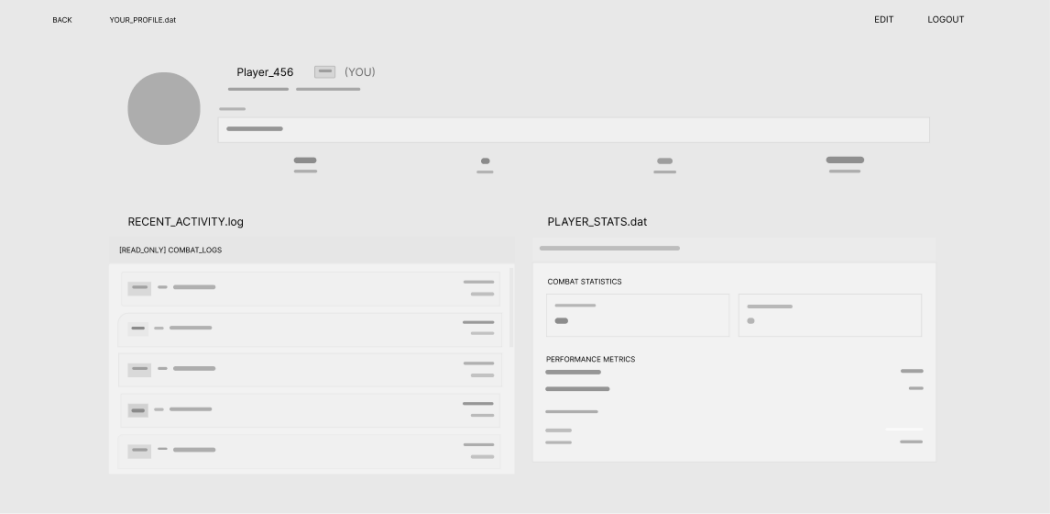
- *Use Case Description*

Allows users to view and edit their profile information, upload an avatar, and view their game statistics.

- *Activity Diagram*

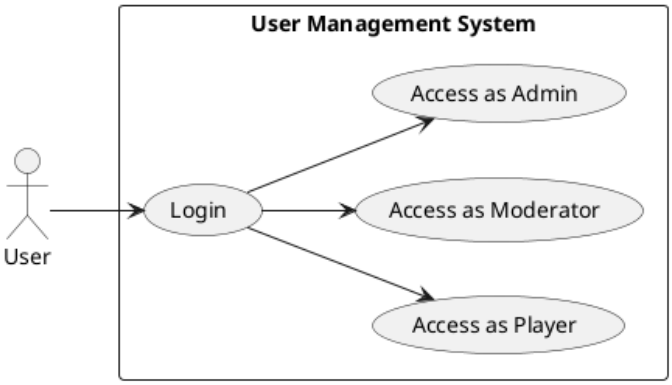


- *Wireframe*



1.4 Role-based access

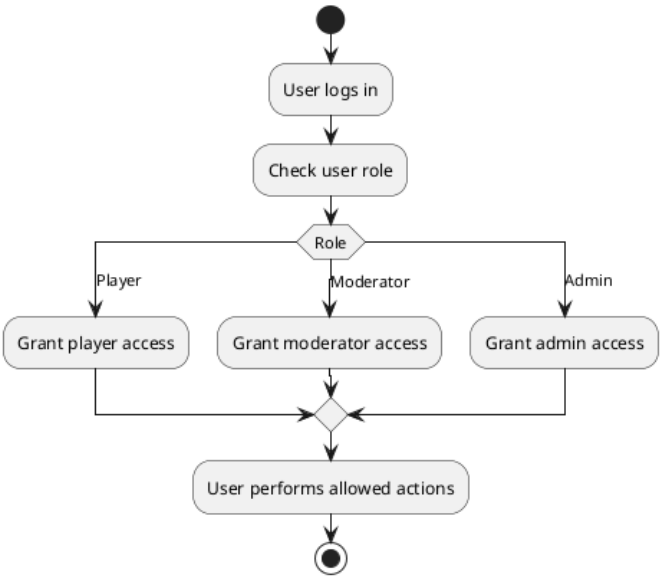
- *Use Case Diagram*



- *Use Case Description*

Grants users access to features and actions based on their assigned role (player, moderator, admin).

- *Activity Diagram*

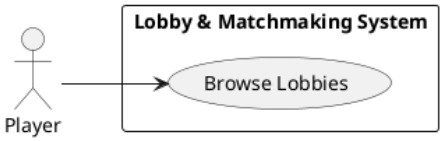


- *Wireframe*

Module 2: Lobby and Matchmaking

2.1 Lobby browser

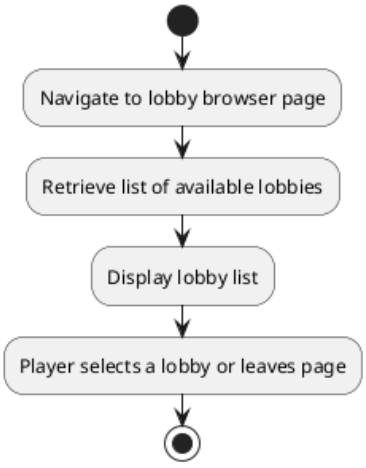
- *Use Case Diagram*



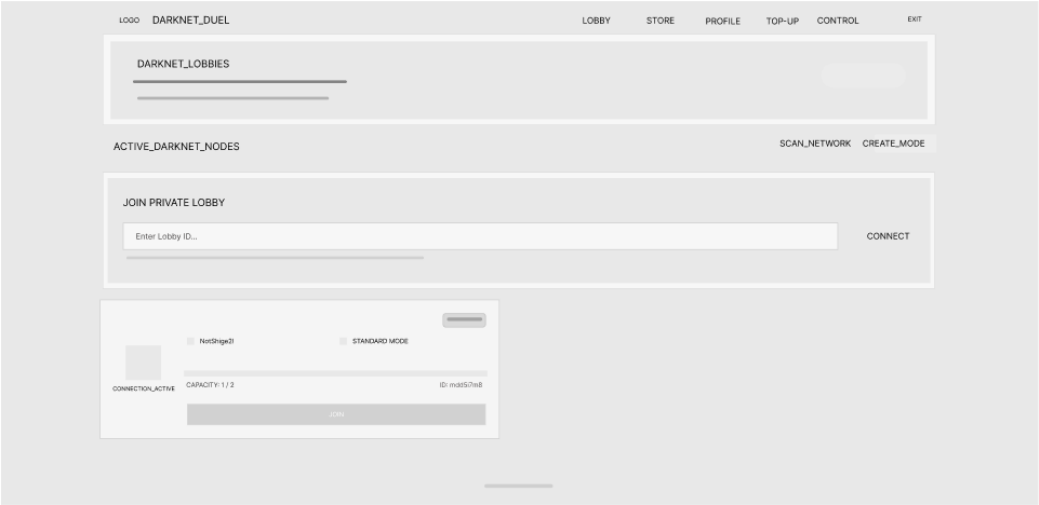
- *Use Case Description*

Allows a player to view a list of available game lobbies. The system retrieves the current lobby list and displays it to the player in real time.

- *Activity Diagram*

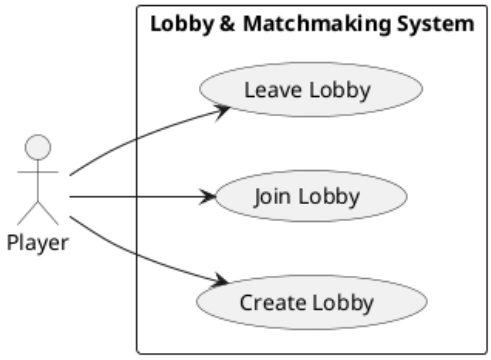


- *Wireframe*



2.2 Create/join/leave lobbies

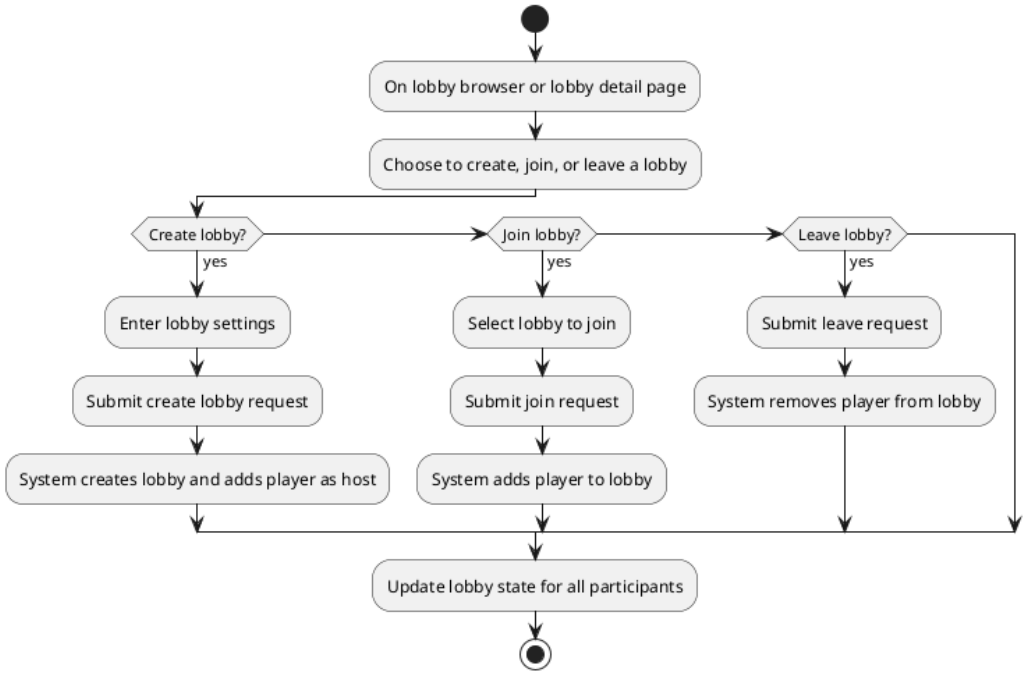
- *Use Case Diagram*



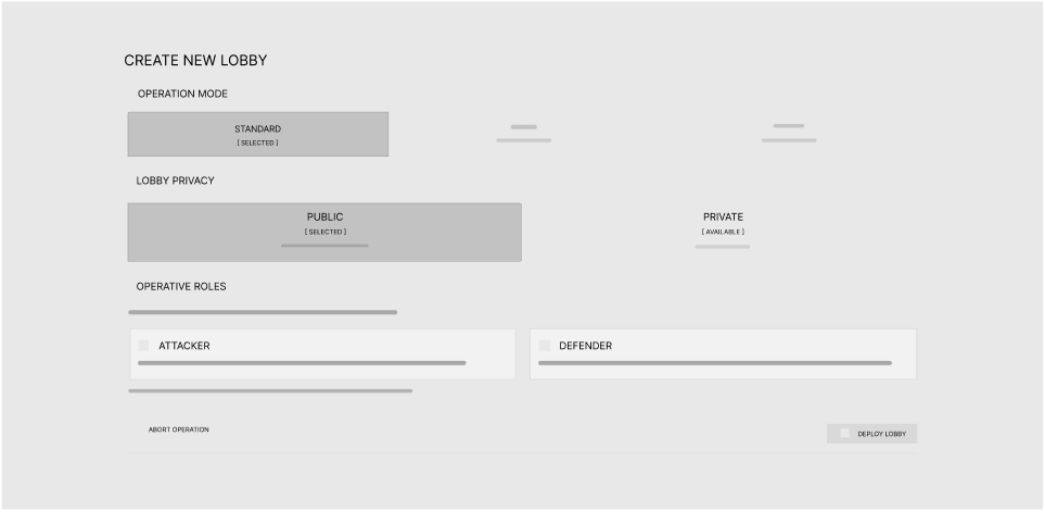
- *Use Case Description*

Allows a player to create a new game lobby, join an existing lobby, or leave a lobby. The system manages lobby creation, player entry/exit, and updates the lobby state for all participants in real time.

- *Activity Diagram*

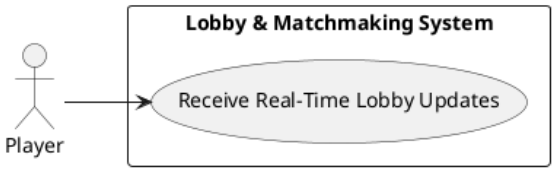


- *Wireframe*



2.3 Real-time lobby updates

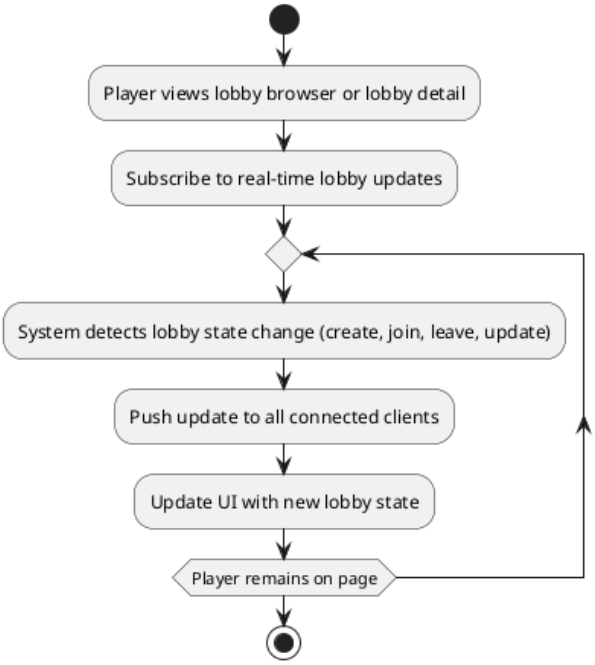
- *Use Case Diagram*



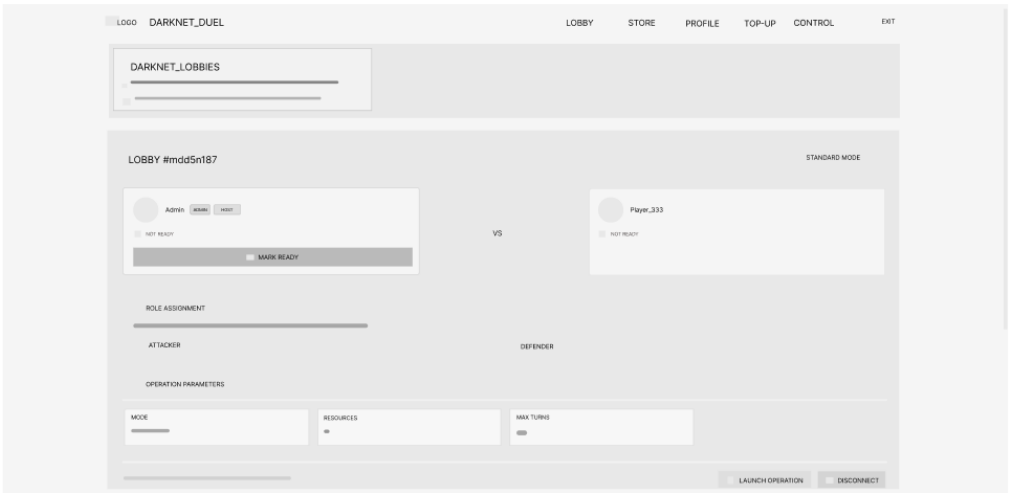
- *Use Case Description*

Ensures that players receive immediate updates when lobbies are created, joined, left, or modified. The system pushes lobby state changes to all connected clients in real time, keeping the lobby browser and details up to date.

- *Activity Diagram*

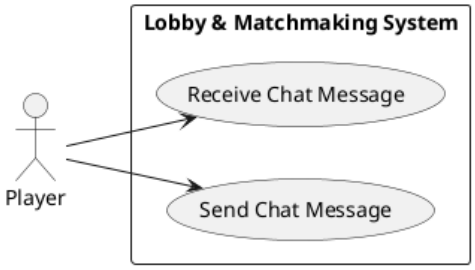


- *Wireframe*



2.4 Lobby chat

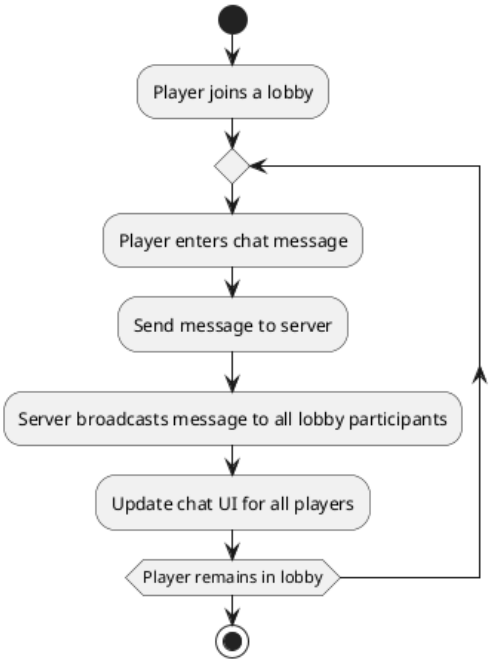
- *Use Case Diagram*



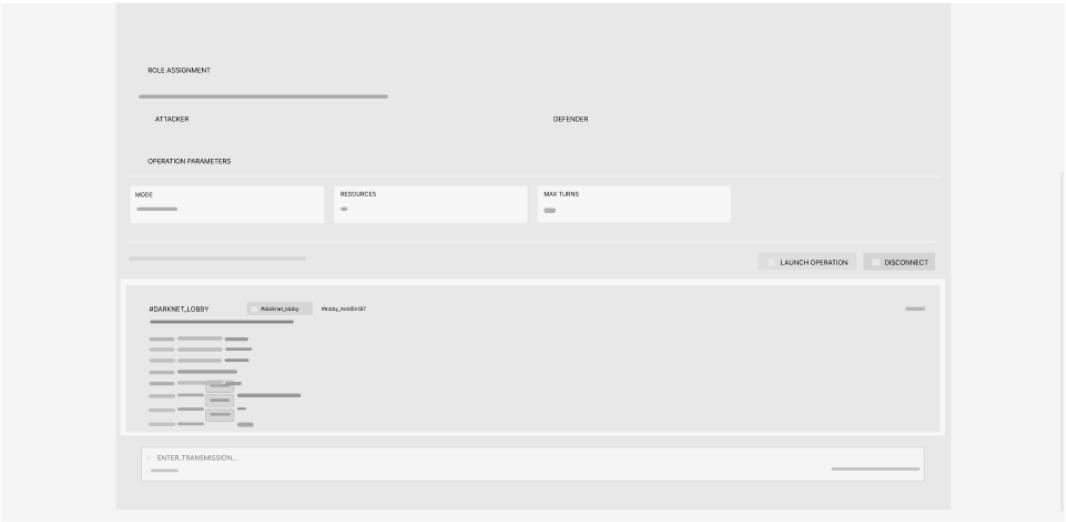
- *Use Case Description*
  Allows players in a lobby to send and receive chat messages in real time. The system broadcasts

messages to all participants in the lobby, enabling communication before the game starts.

- *Activity Diagram*



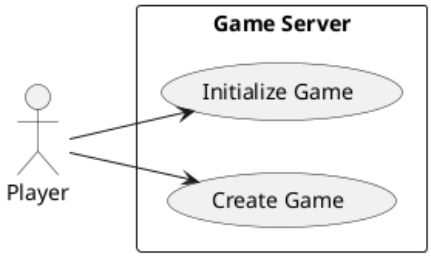
- *Wireframe*



Module 3: Real-Time Multiplayer Game

3.1 Game Creation and initialization

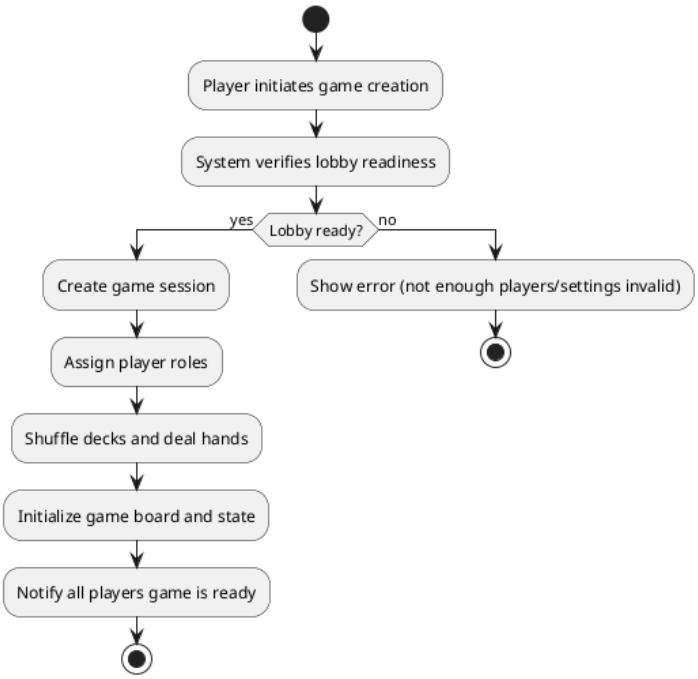
- *Use Case Diagram*



- *Use Case Description*

Allows a player to create a new game session. The system initializes the game state, assigns player roles, shuffles decks, and prepares the game board for play.

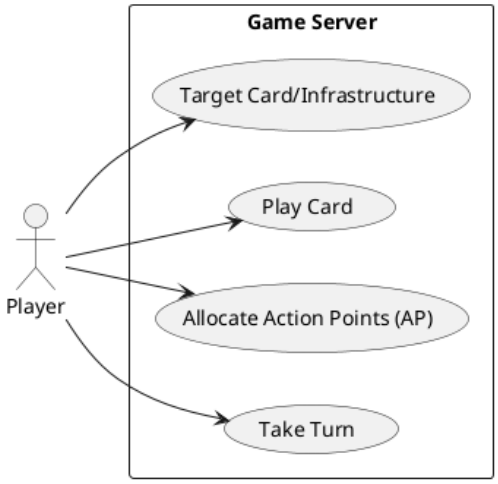
- *Activity Diagram*



- *Wireframe*

3.2 Turn-based gameplay, AP allocation, card play, targeting

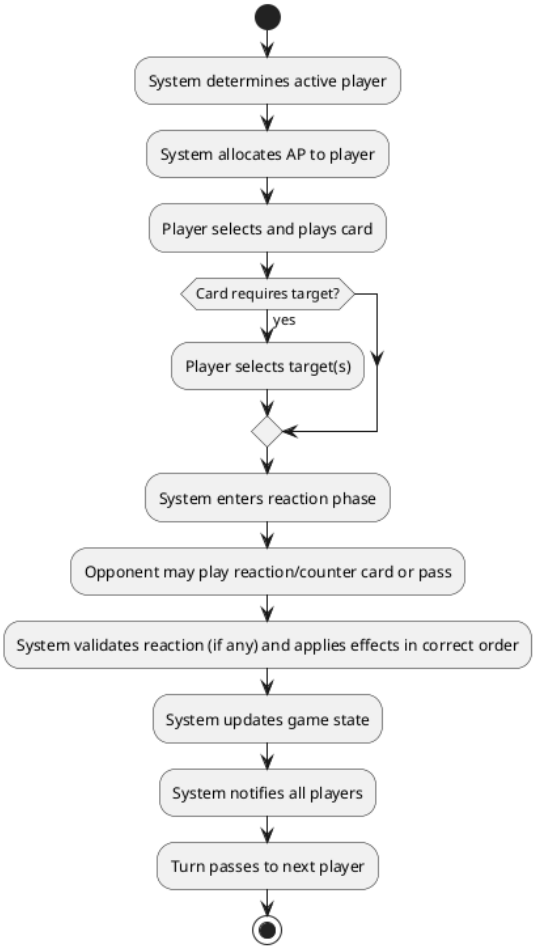
- *Use Case Diagram*



- *Use Case Description*

Allows players to take turns, receive action points (AP), play cards from their hand, select targets for card effects, and handle a reaction phase where the opponent may respond to a card play. The system enforces turn order, validates actions, manages the reaction phase, and updates the game state accordingly.

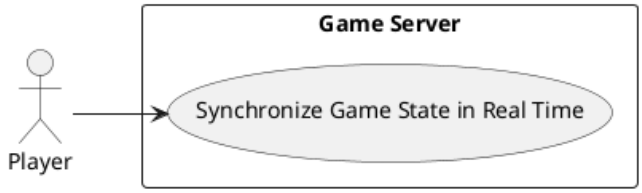
- *Activity Diagram*



- *Wireframe*

3.3 Real-time state synchronization

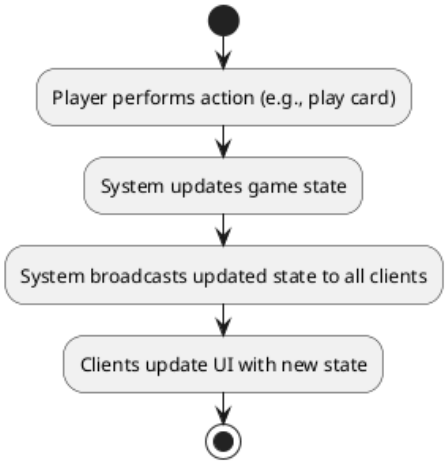
- *Use Case Diagram*



- *Use Case Description*

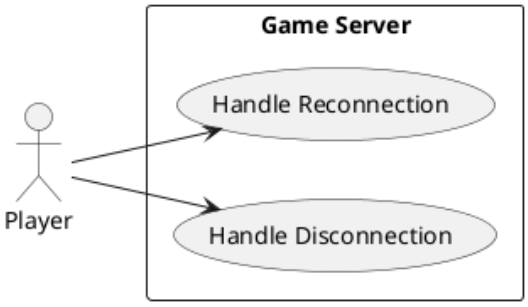
Ensures that all players receive immediate updates to the game state whenever an action occurs. The system uses WebSockets (Socket.IO) and boardgame.io to broadcast state changes to all connected clients in real time.

- *Activity Diagram*



3.4 Disconnection/reconnection handling

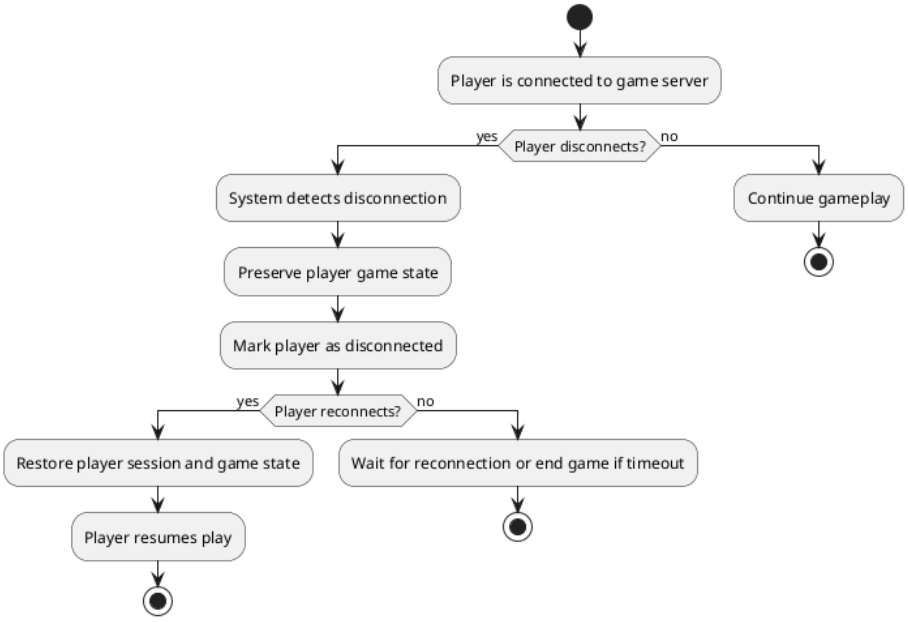
- *Use Case Diagram*



- *Use Case Description*

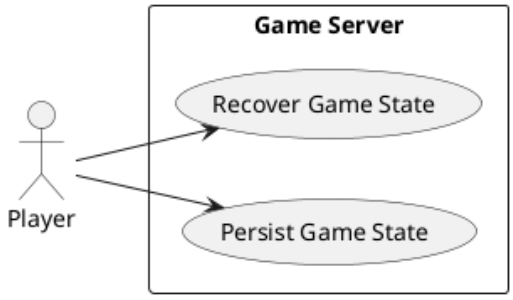
Ensures that if a player disconnects from the game (e.g., due to network issues), the system preserves their game state and allows them to reconnect and resume play without loss of progress.

- *Activity Diagram*



3.5 Game state persistence and recovery

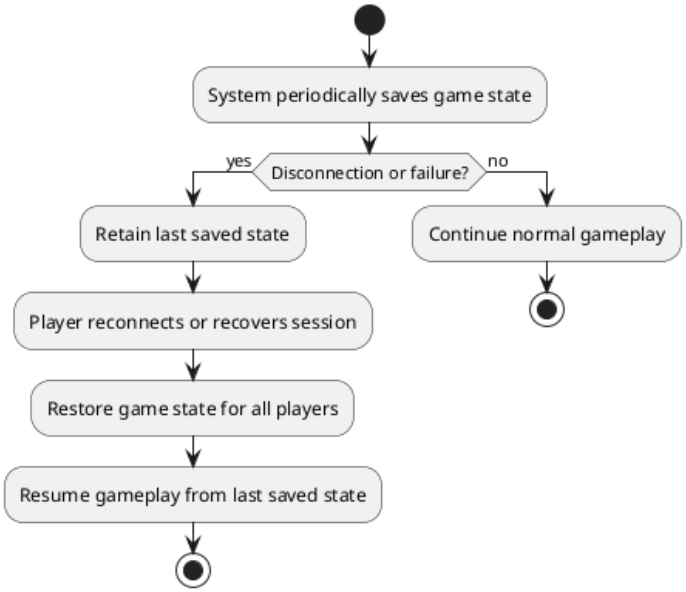
- *Use Case Diagram*



- *Use Case Description*

Ensures that the current game state is regularly saved and can be recovered in case of disconnection, refresh, or server failure. The system persists game data and allows players to resume from the last saved state.

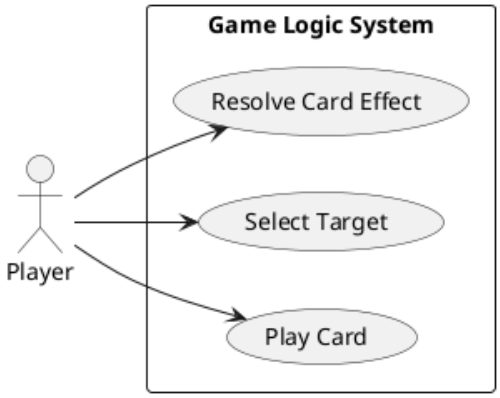
- *Activity Diagram*



Module 4: Card System and Game Logic

4.1 Card play, targeting, and effect resolution

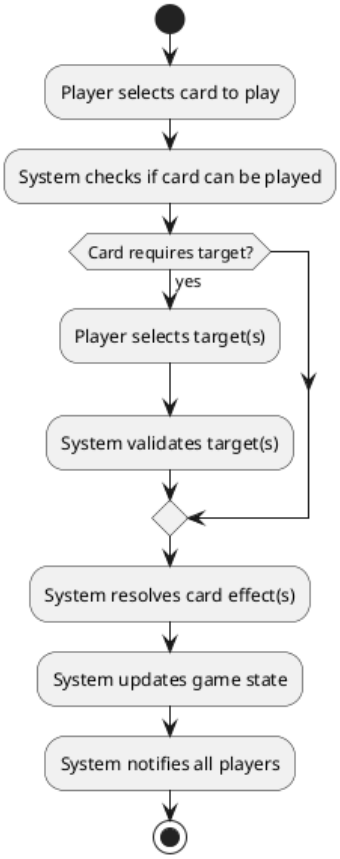
- *Use Case Diagram*



- *Use Case Description*

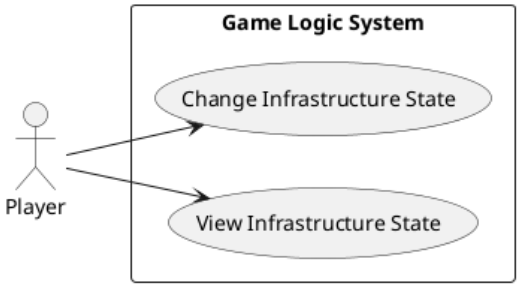
Allows a player to play a card from their hand, select valid targets (if required), and have the system resolve the card's effects according to game rules. The system validates the action, applies effects, and updates the game state.

- *Activity Diagram*



4.2 Infrastructure state tracking

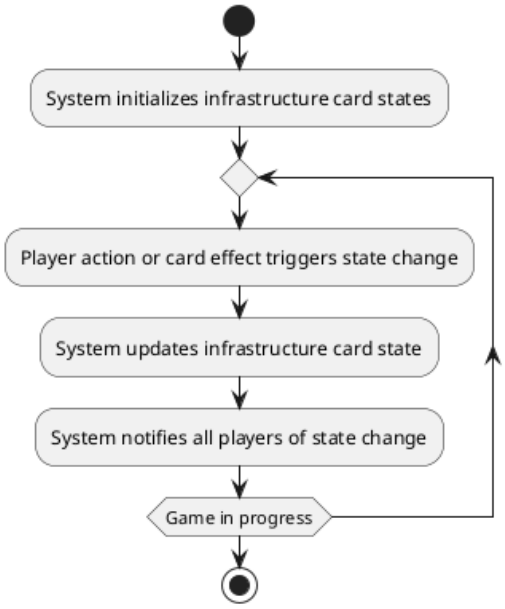
- *Use Case Diagram*



- *Use Case Description*

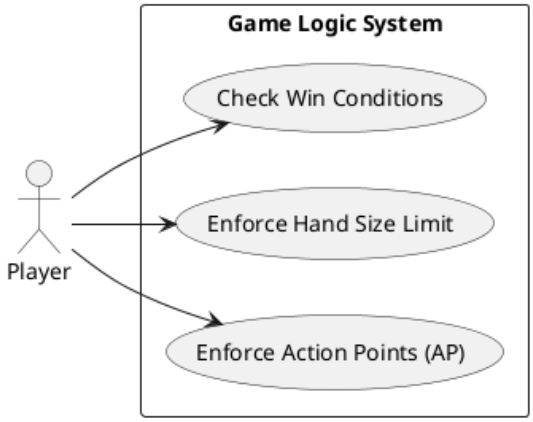
Allows the system to track and update the state of infrastructure cards (e.g., secure, vulnerable, compromised, fortified) as a result of player actions and card effects. Players can view the current state of all infrastructure cards.

- *Activity Diagram*



4.3 Game rules enforcement

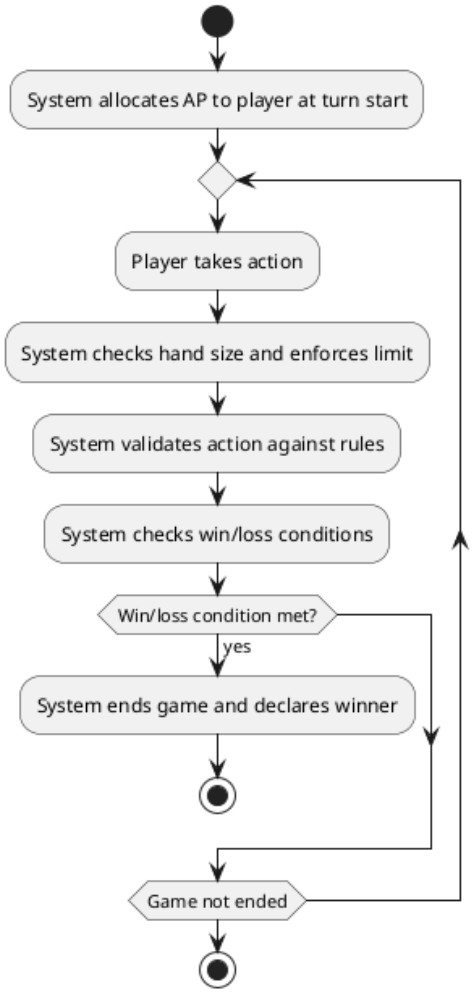
- *Use Case Diagram*



- *Use Case Description*

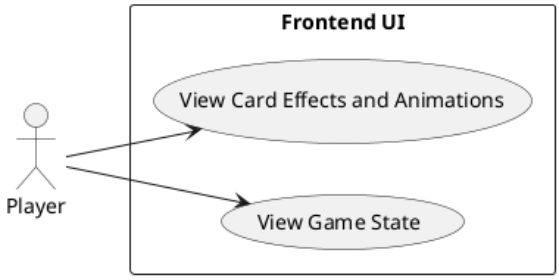
Ensures that all game rules are enforced, including action point (AP) allocation, hand size limits, and win/loss conditions. The system validates player actions and ends the game when win conditions are met.

- *Activity Diagram*



4.4 Game state visualization

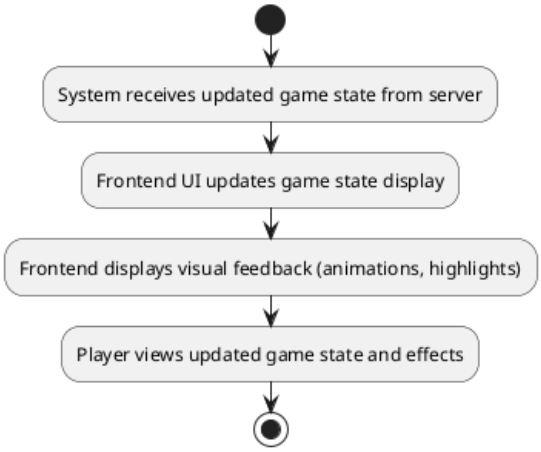
- *Use Case Diagram*



- *Use Case Description*

Allows players to view the current game state, including cards in play, player hands, infrastructure states, and ongoing effects, through the frontend UI. The system provides real-time updates and visual feedback (e.g., animations, highlights) for all game actions.

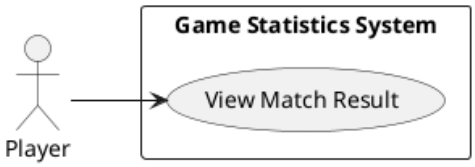
- *Activity Diagram*



Module 5: Game Statistics and Result Tracking

5.1 Match result display

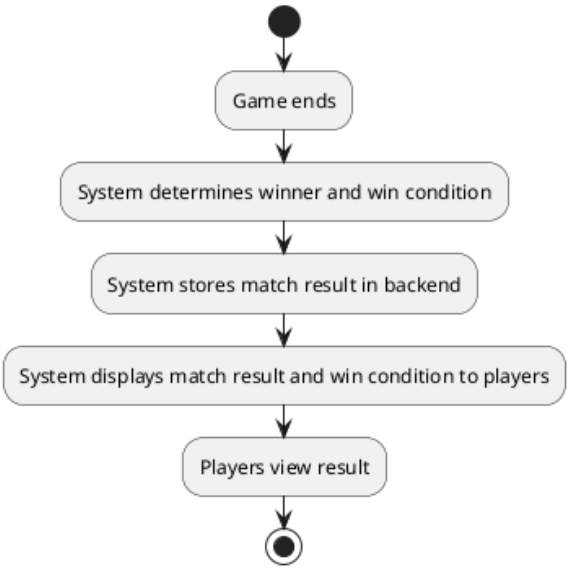
- *Use Case Diagram*



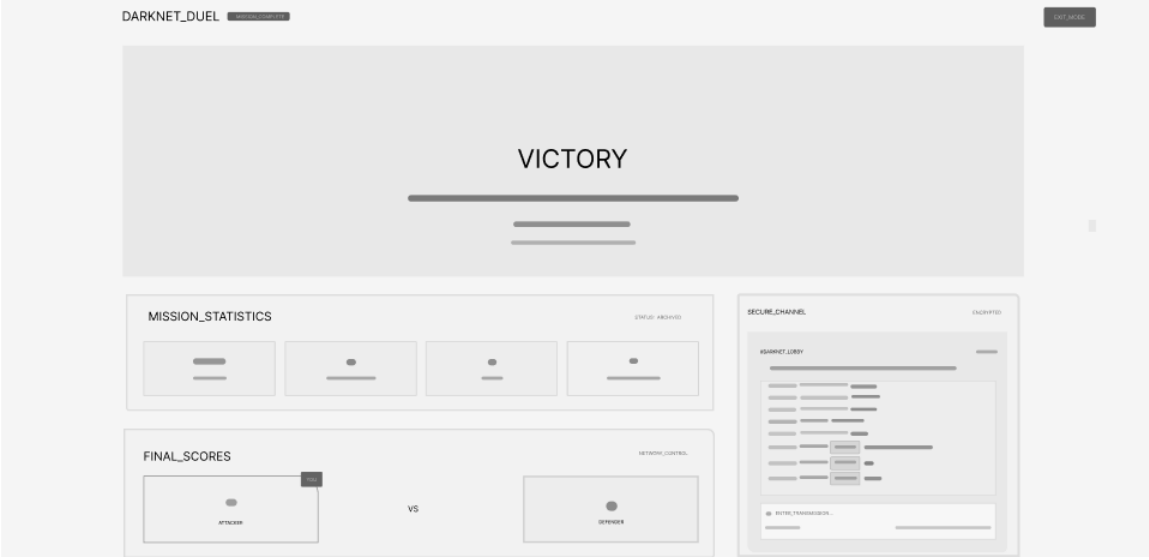
- *Use Case Description*

Allows players to view the outcome of a completed match, including victory or defeat and the specific win condition. The system displays the result immediately after the game ends.

- *Activity Diagram*

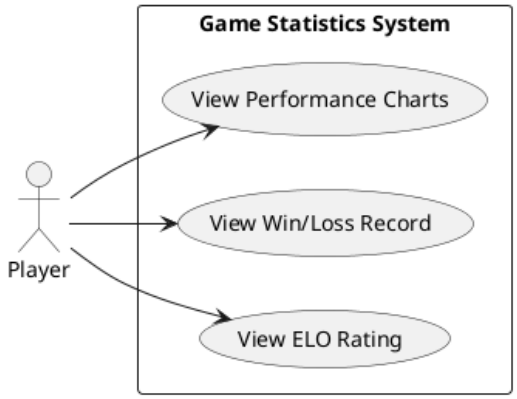


- *Wireframe*



5.2 Player performance statistics

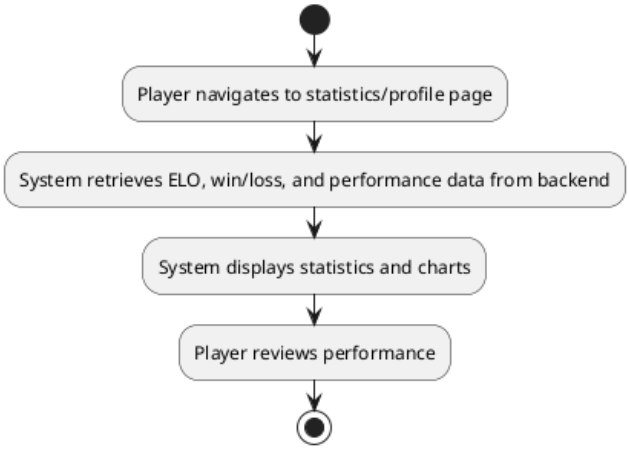
- *Use Case Diagram*



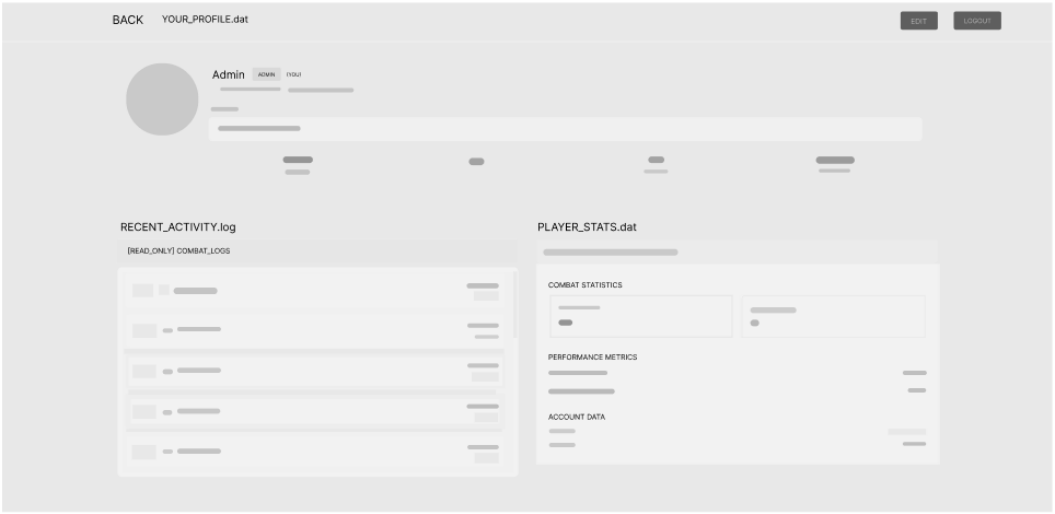
- *Use Case Description*

Allows players to view their performance statistics, including ELO rating, win/loss record, and visual charts of their progress over time. The system retrieves and displays up-to-date statistics from the backend.

- *Activity Diagram*

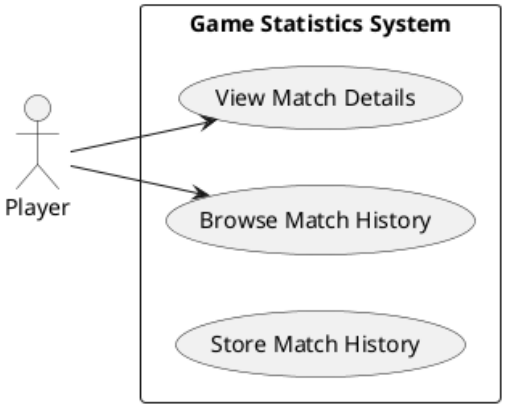


- *Wireframe*



5.3 Match history storage and browsing

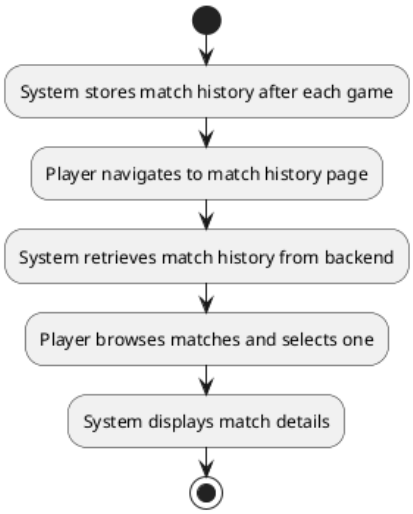
- *Use Case Diagram*



- *Use Case Description*

The system stores completed match history and enables players to browse their past matches and view detailed information about each match. The system retrieves and displays match history from the backend.

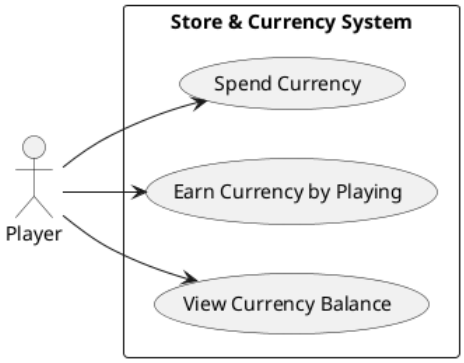
- *Activity Diagram*



Module 6: Store and Currency

6.1 In-game currency management

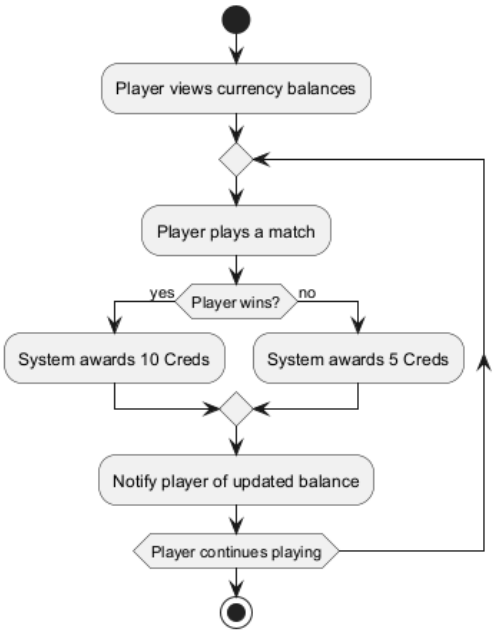
- *Use Case Diagram*



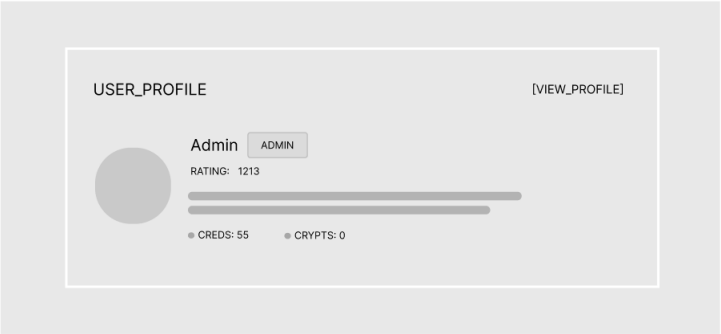
- *Use Case Description*

Allows players to view their in-game currency balances (Creds, Crypts), earn Creds by playing matches (10 for a win, 5 for a loss), and spend currency on purchases or other in-game actions. The system updates balances and enforces currency rules.

- *Activity Diagram*

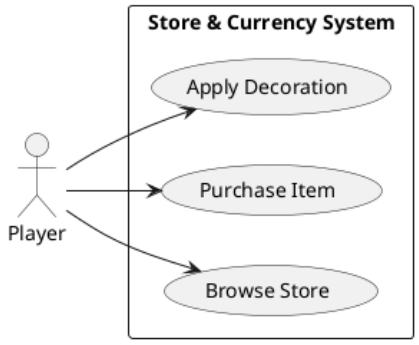


- *Wireframe*



6.2 Store browsing, item purchase, and application of decoration

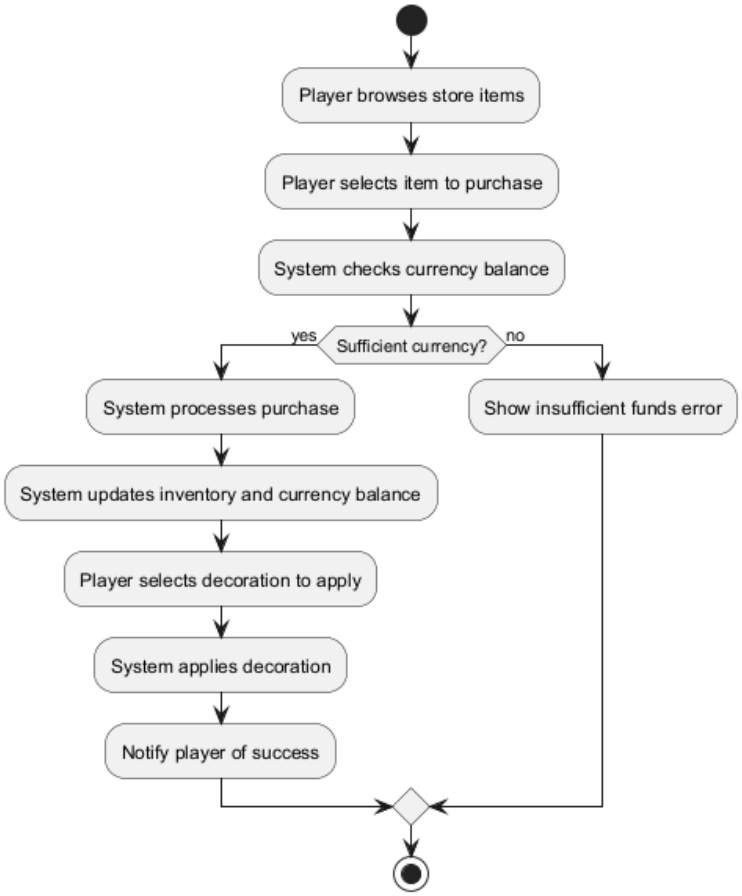
- *Use Case Diagram*



- *Use Case Description*

Allows players to browse the in-game store, purchase items using in-game currency, and apply purchased decorations to their profile or game elements. The system validates purchases, updates inventory, and applies decorations.

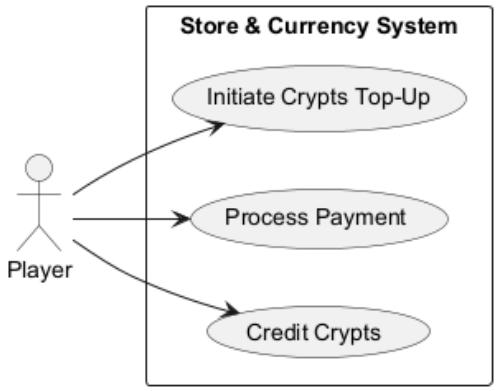
- *Activity Diagram*



- *Wireframe*

6.3 Payment integration

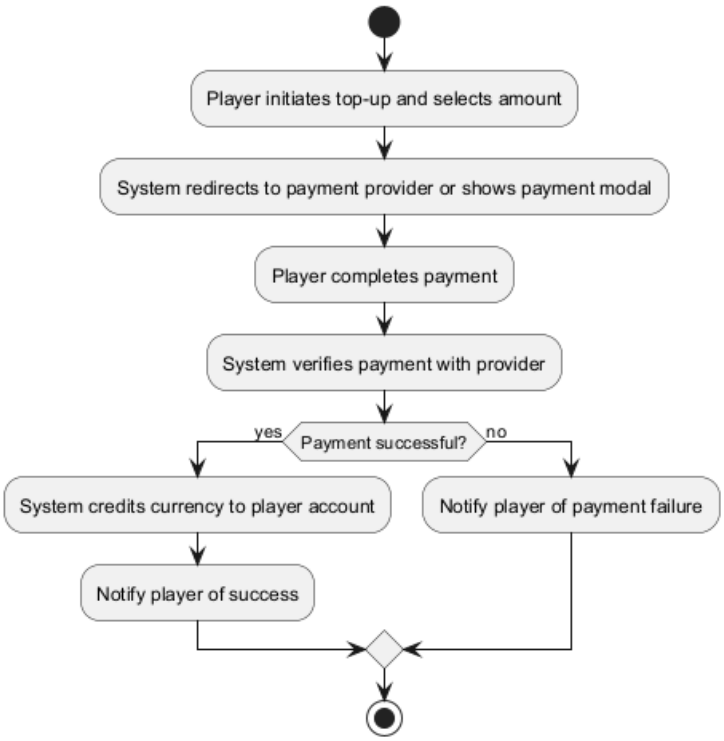
- *Use Case Diagram*



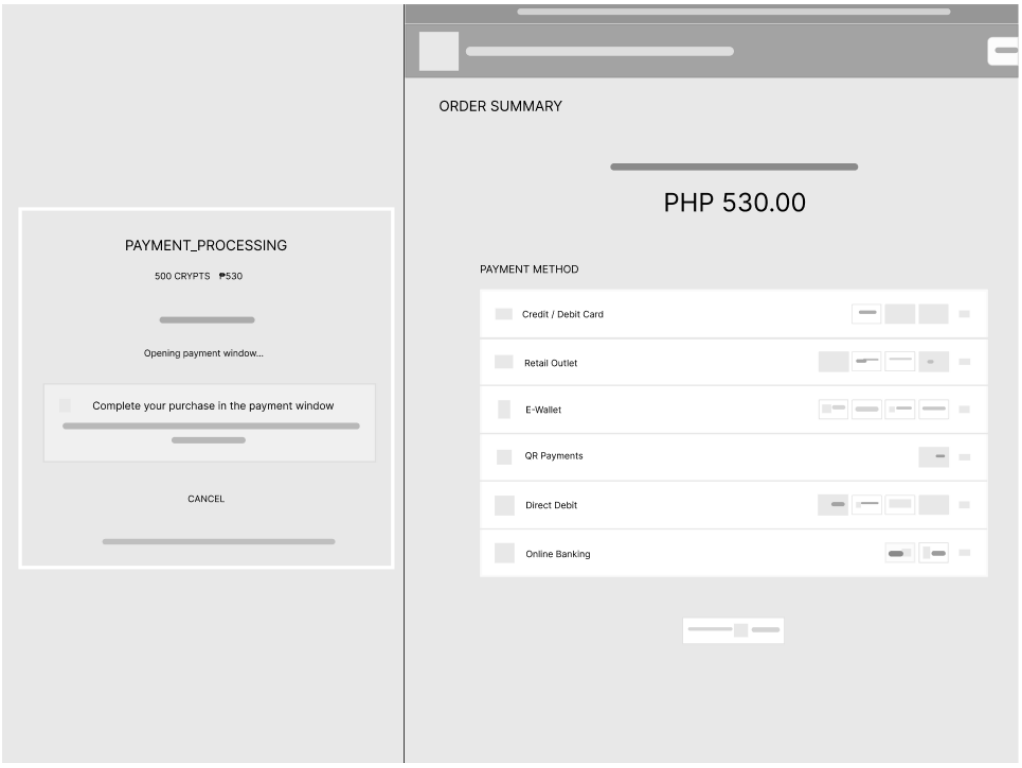
- *Use Case Description*

Allows players to purchase in-game currency (Crypts) using real money through integrated payment providers. The system processes payments, credits currency to the player's account, and notifies the player of the result.

- *Activity Diagram*



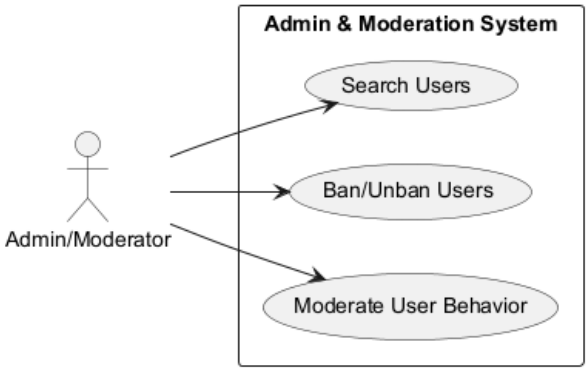
- *Wireframe*



Module 7: Admin and Moderation Tools

7.1 User search, ban, and moderation

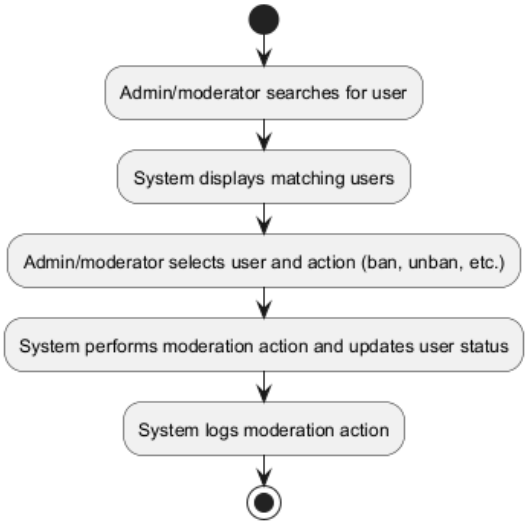
- *Use Case Diagram*



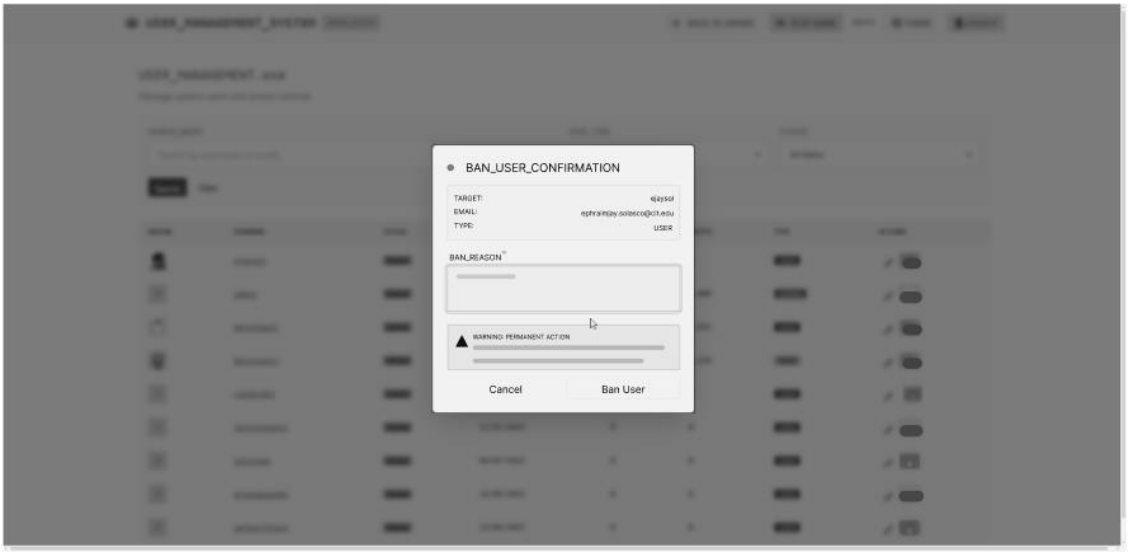
- *Use Case Description*

Allows admins and moderators to search for users, ban or unban users, and moderate user behavior (e.g., issue bans). The system enforces access control and logs all moderation actions.

- *Activity Diagram*

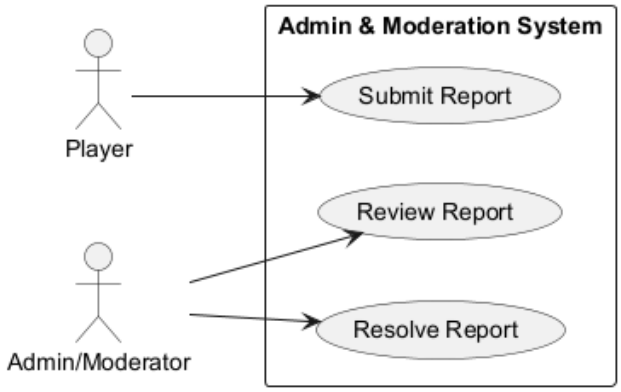


- *Wireframe*



7.2 Report management

- *Use Case Diagram*



- *Use Case Description*

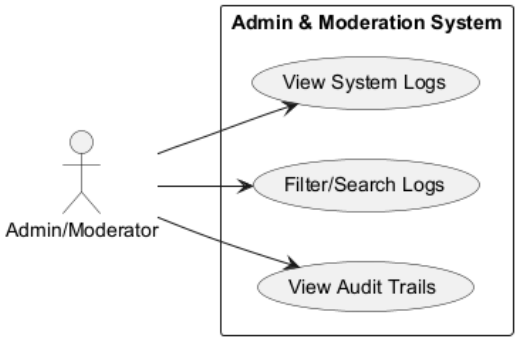
Allows players to submit reports about inappropriate behavior or issues, and allows admins/moderators to review and resolve these reports. The system tracks report status and notifies relevant parties of updates.

- *Activity Diagram*

- *Wireframe*

7.3 System logs and audit logs

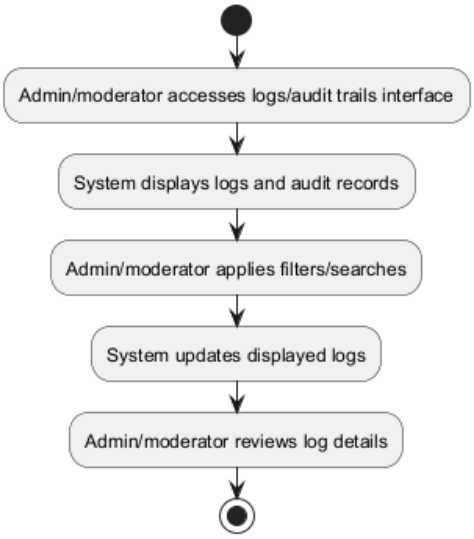
- *Use Case Diagram*



- *Use Case Description*

Allows admins and moderators to view, filter, and search system logs and audit trails for security, moderation, and troubleshooting purposes. The system records all relevant actions and provides tools for log analysis.

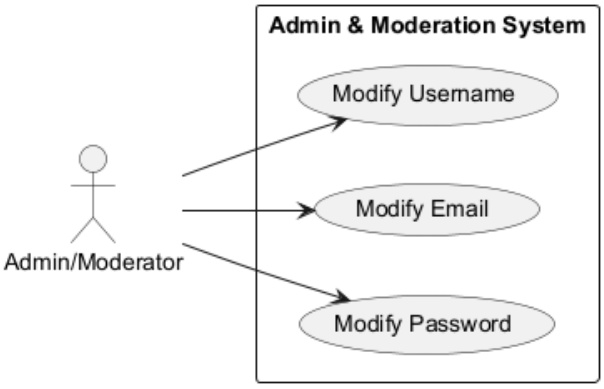
- *Activity Diagram*



- *Wireframe*

7.4 User modification

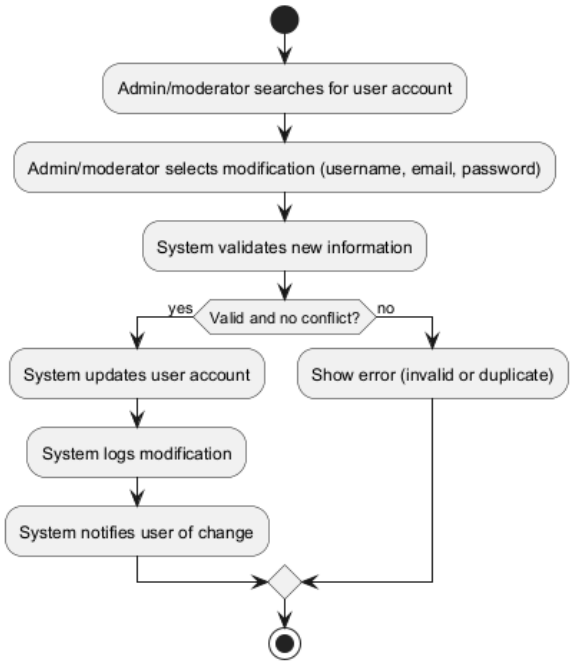
- *Use Case Diagram*



- *Use Case Description*

Allows admins and moderators to modify user account details, including username, email, and password, for support or security reasons. The system enforces access control and logs all modifications.

- *Activity Diagram*

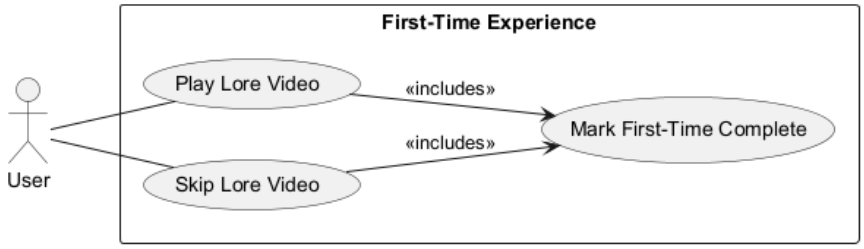


- *Wireframe*

Module 8: First-Time Experience

8.1 Lore video playback for first-time users on initial login

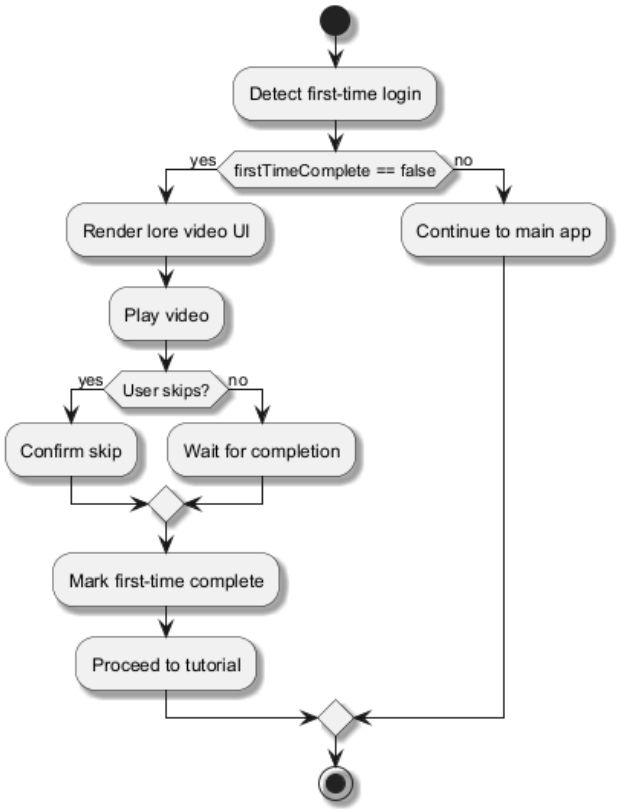
- *Use Case Diagram*



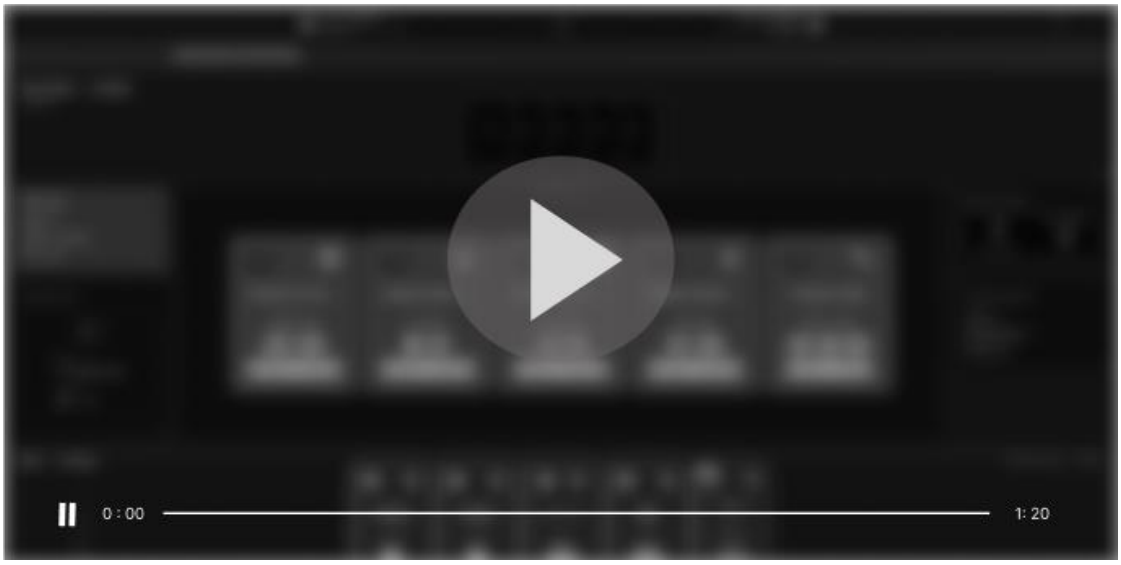
- *Use Case Description*

Allows admins and moderators to search for users, ban or unban users, and moderate user behavior (e.g., issue bans). The system enforces access control and logs all moderation actions.

- *Activity Diagram*

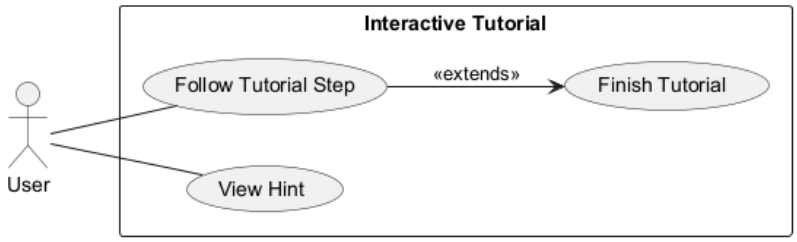


- *Wireframe*



7.2 Interactive tutorial guiding new players through core mechanics

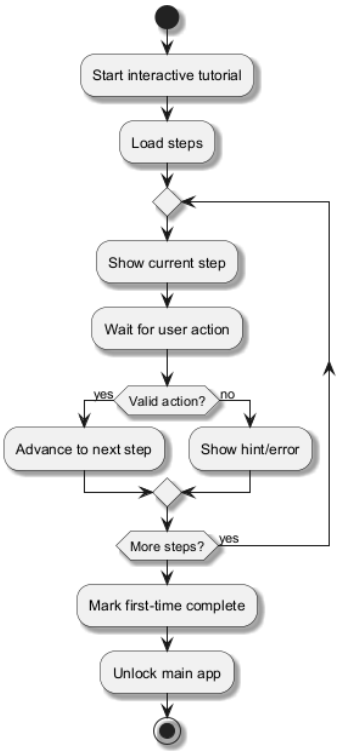
- *Use Case Diagram*



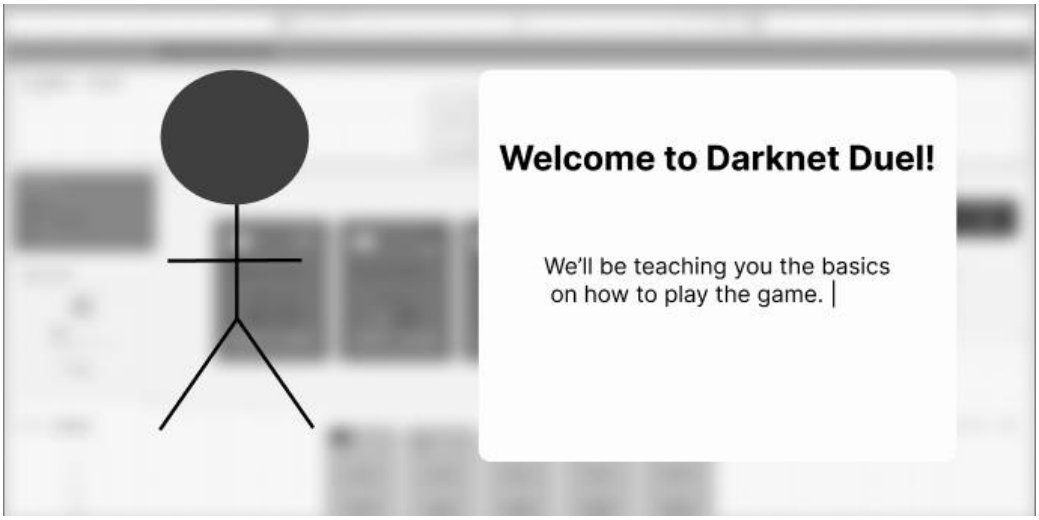
- *Use Case Description*

Allows players to submit reports about inappropriate behavior or issues, and allows admins/moderators to review and resolve these reports. The system tracks report status and notifies relevant parties of updates.

- *Activity Diagram*



- *Wireframe*



3.4 Non-functional requirements

Performance

- Card action resolution < 1s
- Game initialization < 3s
- Max 2s latency for real-time actions
- Support for 1000+ concurrent users
- Database queries < 500ms

Security

- JWT authentication, HTTPS everywhere
- Input validation, XSS/CSRF/SQLi protection
- Server-side validation of all game actions
- Rate limiting and abuse prevention
- Secure WebSocket connections
- Audit logging for administrative actions
- Security Headers (Helmet) and strict CORS policies
- GDPR features (export/deletion)
- Session timeout and enforced automatic logout
- CI/CD security gate: fail builds on detected vulnerabilities

Reliability

- 99.9% uptime
- Automatic reconnection for disconnects
- Game state persistence on refresh
- Regular database backups
- Graceful error handling
- In-game offline handling with 20s grace period before stopping session