EyAy

Ozan Özak - 87242

Kaya Yaylalı - 82899

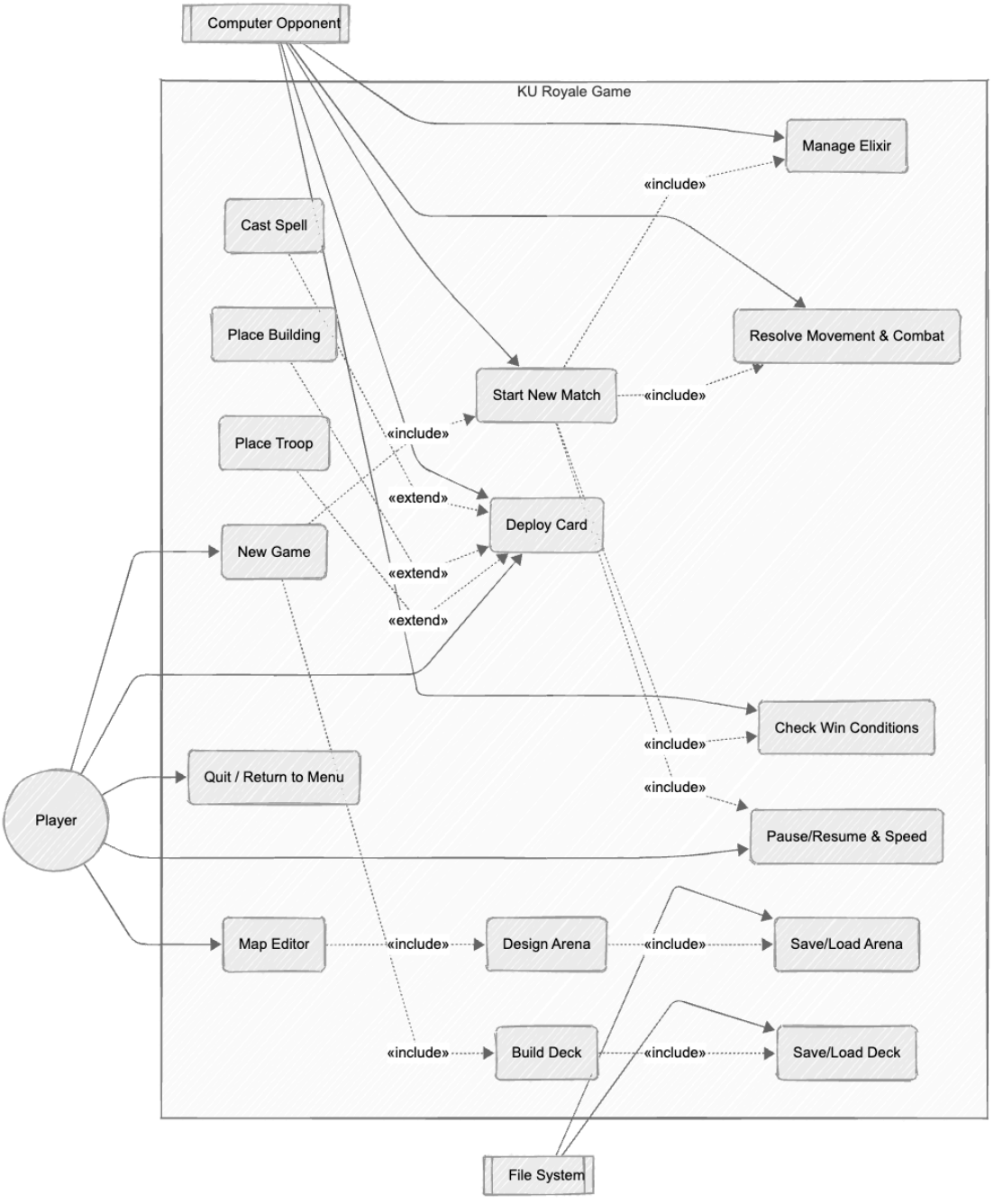
Deniz Ünal - 87041

Demir İyigün - 87534

Ege Töman - 87169

Yiğit Tutaş - 83482

Use Case Diagram



Use Case Narratives

Use Case Name: Design Arena
Use Case ID: UC-01
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Goal: Let the player design and save an arena by placing 3 towers (2 Crown, 1 King) and 1–3 bridges; the AI receives a mirrored layout.
Stakeholders & Interests:

- **Player:** Wants control over battlefield layout; expects invalid placements (e.g., on bridges) to be blocked.
- **Game System:** Must validate placements, mirror the layout for AI, and persist the arena.

Preconditions:

- Map Editor is opened from the main menu.
- An editable arena canvas is displayed.

Postconditions:

- A valid arena layout (towers + bridges) is stored; will be used for future matches.

Main Success Scenario:

1. Player opens Map Editor.
2. System shows empty/mutable arena grid with a river.
3. Player places 2 Crown Towers + 1 King Tower on own side.
4. Player places 1–3 bridges.
5. System validates (no buildings/towers on bridges; all on player side).
6. Player clicks Save; system persists the arena and generates AI's mirrored layout.

Extensions:

3a. **Invalid tower/bridge placement:**

- Show red overlay/error; disallow drop.

5a. **Save failure:**

- Show error; allow retry.

6a. **Player cancels:**

- Discard changes and return to menu.

Use Case Name: Build Deck
Use Case ID: UC-02
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Goal: Choose eight unique cards from the catalog (troops, buildings, spells) and save the deck.

Preconditions:

- **Card catalog is loaded (troops, buildings, spells are available to browse).**
- **The player has access to at least 8 cards that can be used in a deck.**

Main Success Scenario

1-Player navigates to the Deck Builder screen.

2-System displays:

- **The full card catalog (troops, buildings, spells),**
- **Current empty deck slots (or existing deck if editing),**
- **Basic stats for each card (e.g., elixir cost, type, rarity).**

3-Player selects a card to add to the deck.

4-System validates that the selected card is not already in the deck and adds it to one of the available deck slots, updating any displayed deck stats (e.g., average elixir).

5-Steps 3–4 repeat until the player has chosen 8 unique cards.

6- Player clicks Save Deck.

7-The system confirms that the deck is saved.

Extensions:

2a. **Duplicate chosen:**

- Show "No duplicates allowed."

3a. **Not exactly eight:**

- Show validation error.

4a. **Save failure:**

- Show error; allow retry.

Post-conditions:

- A deck containing exactly 8 unique cards is saved for the player.
- The saved deck is associated with the player's profile and can be used in battles.

Use Case Name: Start New Match
Use Case ID: UC-03
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Secondary Actor: Computer Opponent
Goal: Initialize a 3-minute match using the saved arena and decks; set up timers, elixir, hands, and towers.
Stakeholders & Interests:

- **Player:** Wants a smooth transition into battle with correct settings.
- **Game System:** Must correctly initialize all match components.

Preconditions:

- Valid arena and deck exist (or defaults available).
- Player selected New Game.

Postconditions:

- Match is active; timer, elixir, hands, and UI initialized for both sides.

Main Success Scenario:

1. System loads arena (mirrored for AI) and player deck; assigns AI deck.
2. Initializes 3:00 timer; sets elixir to 5 (max 10).
3. Draws 4-card hands for player and AI.
4. Enters battle screen with cards, elixir bar, timer, health bars, pause and speed controls.

Extensions:

1a. **Missing arena/deck:**

- Load defaults.

2a. **Final minute:**

- Switch to Double Elixir.

4a. **Pause pressed:**

- Freeze updates; show overlay.

Use Case Name: Deploy Card
Use Case ID: UC-04
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Secondary Actor: Computer Opponent
Goal: Spend elixir to play a card from hand; placement restricted to player side for troops/buildings; spells can target anywhere. Hand cycles afterward.
Stakeholders & Interests:

- **Player:** Wants responsive deployment and clear feedback on invalid placement.
- **Game System:** Must validate costs and placements and update hand state.

Preconditions:

- Actor has enough elixir for the selected card.
- A valid placement or target is chosen.

Postconditions:

- Elixir decreases by card cost; effect created; new card drawn into hand.

Main Success Scenario:

1. Actor selects a card in hand.
2. System checks elixir sufficiency; otherwise blocks.
3. Actor clicks a valid cell or area.
4. System places the effect (troop, building, or spell), deducts elixir, and draws the next card.

Extensions:

2a. **Not enough elixir:**

- Show disabled state or message.

3a. **Invalid cell:**

- Block placement and highlight red.

Use Case Name: Place Troop
Use Case ID: UC-05
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Secondary Actor: Computer Opponent
Goal: Spawn a troop that moves and attacks automatically according to its stats and targeting rules.
Stakeholders & Interests:

- **Player:** Wants strategic control and predictable behavior.
- **Game System:** Must simulate movement and combat accurately.

Preconditions:

- Deploy Card preconditions are met.
- Placement is on player side.

Postconditions:

- Troop participates in movement and combat simulation.

Main Success Scenario:

1. Player places a troop (e.g., Knight, Musketeer, Minions).
2. System spawns troop with HP, damage, range, speed, and target types.
3. Troop auto-moves toward nearest enemy/tower; ground units cross bridges; air units fly directly.
4. Combat resolves according to stats.

Extensions:

2a. **Special targeting:**

- Giant and Hog Rider target only buildings/towers.

3a. **No direct path:**

- Ground unit routes to the nearest bridge.

Use Case Name: Place Building
Use Case ID: UC-06
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Secondary Actor: Computer Opponent
Goal: Place a stationary defensive/spawner/special building that acts for its lifetime.
Stakeholders & Interests:

- **Player:** Wants tactical placement and clear lifetime feedback.
- **Game System:** Must simulate range, damage, and expiration correctly.

Preconditions:

- Deploy Card preconditions are met.
- Placement is on player side and not on a bridge.

Postconditions:

- Building acts until destroyed or expiration; spawners and collectors tick as specified.

Main Success Scenario:

1. Player places a building (e.g., Cannon, Tesla, Tombstone, Elixir Collector).
2. System spawns the building with range/damage, spawn cadence, or elixir production behavior.
3. Building acts each tick; some self-destruct at end of lifetime.

Extensions:

1a. **Invalid on-bridge placement:**

- Block and warn.

2a. **Elixir Collector:**

- Produces periodic elixir and nets positive if it survives.

Use Case Name: Cast Spell
Use Case ID: UC-07
Scope: KU Royale – Phase 1
Level: User goal
Primary Actor: Player
Secondary Actor: Computer Opponent
Goal: Apply an instant area effect at a target location (e.g., Zap, Arrows, Fireball, Rocket).
Stakeholders & Interests:

- **Player:** Wants precise targeting and timely effects.

- **Game System:** Must apply AoE damage and status accurately.

Preconditions:
- Deploy Card preconditions are met.
- A valid area is selected.

Postconditions:
- AoE damage/effects applied; reduced damage to towers when applicable.

Main Success Scenario:
1. Player selects a spell and targets an area.
2. System applies radius damage/effects (stun, splash, etc.).
3. Affected entities update HP; destroyed ones are removed.

Extensions:

2a. **No enemy in area:**
- Spell resolves with no impact.

Use Case Name: Manage Elixir
Use Case ID: UC-08
Scope: KU Royale – Phase 1
Level: Subfunction (included by Start New Match)
Primary Actor: Player
Secondary Actor: Computer Opponent
Goal: Maintain elixir resource: start at 5, cap at 10, refill over time, and double refill in the final minute.

Stakeholders & Interests:
- **Player:** Wants steady resource replenishment and visible status.
- **Game System:** Must track timing and apply limits consistently.

Preconditions:
- Match is running.

Postconditions:
- Elixir values updated; UI shows current elixir; deployments enabled/blocked accordingly.

Main Success Scenario:
1. Each tick, elixir refills toward 10.
2. During the final minute, refill rate doubles.
3. UI continuously displays current elixir.

Extensions:

1a. **Elixir Collector present:**
- Add periodic bonus elixir.

Use Case Name: Resolve Movement & Combat
Use Case ID: UC-09
Scope: KU Royale – Phase 1
Level: Subfunction (included by Start New Match)
Primary Actor: Game Loop
Goal: Update positions and resolve targeting, attacks, and AoE each tick; update health bars..

Stakeholders & Interests:
- **Player:** Wants realistic and predictable combat resolution.
- **Game System:** Ensures fair damage and timing simulation.

Preconditions:
- At least one entity is on the field.

Postconditions:

- Entities move/attack; HP and health bars update; destroyed entities are removed.

Main Success Scenario:

1. For each entity, select a valid target in range respecting target type rules.
2. If attack cooldown is ready, apply damage or area damage.
3. Update HP; remove destroyed entities.
4. Move troops along paths (bridges for ground; direct for air).

Extensions:

1a. **Buildings-only attackers (Giant, Hog Rider):**

- Enforce targeting constraint.

2a. **AoE units (Valkyrie, Wizard, Bomber):**

- Apply splash rules.

Use Case Name: Check Win Conditions
Use Case ID: UC-10
Scope: KU Royale – Phase 1
Level: Subfunction (included by Start New Match)
Primary Actor: Game System
Goal: End the match when a King Tower is destroyed or decide a winner at time-up; tiebreak via rapid HP drain if needed.

Stakeholders & Interests:

- **Player:** Wants fair and clear match outcome.
- **Game System:** Must apply victory logic accurately and display results.

Preconditions:

- Match is ongoing.

Postconditions:

- Winner is determined; match ends; result screen or menu is shown.

Main Success Scenario:

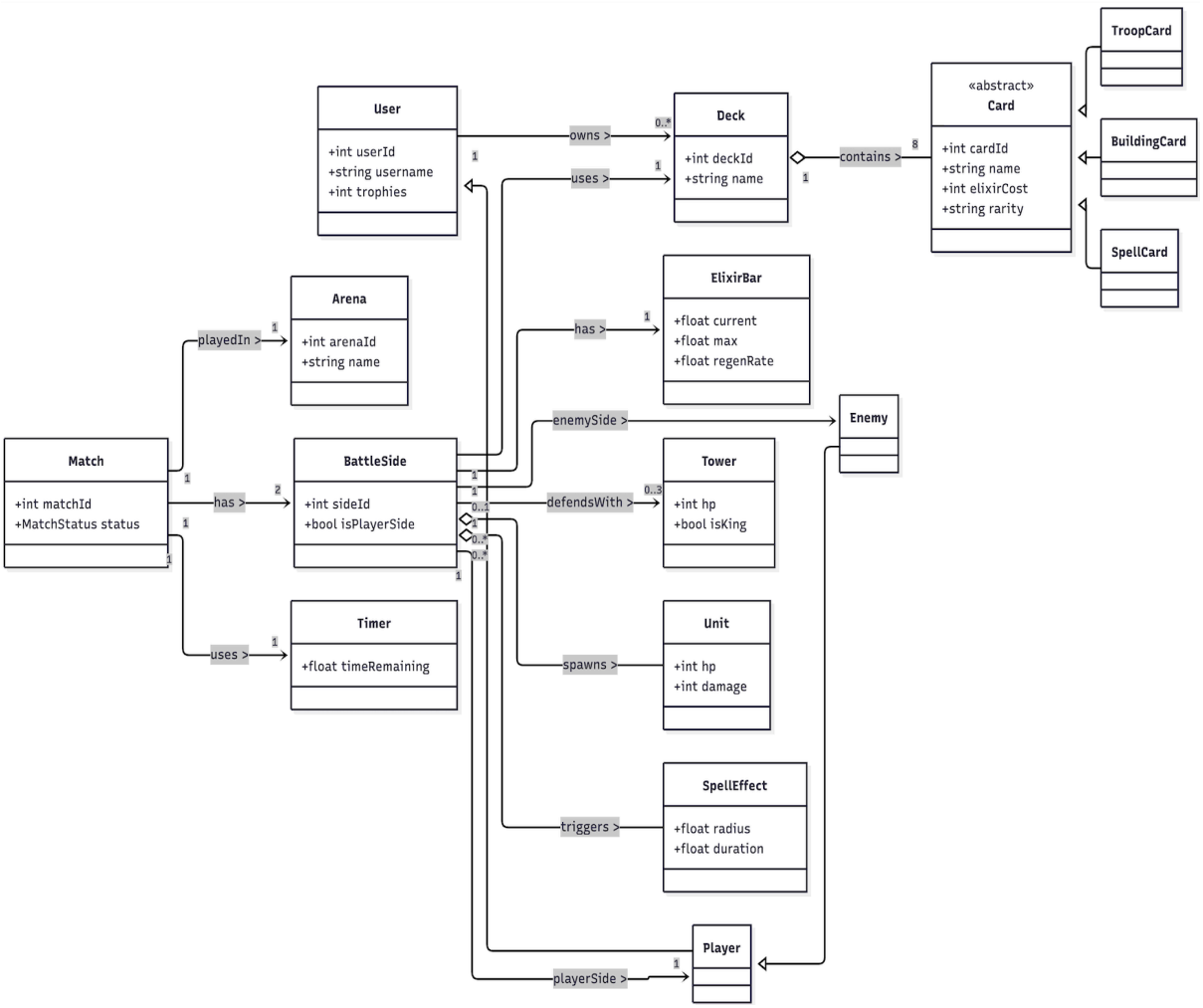
1. If any King Tower reaches 0 HP → end match; other side wins.
2. At 3:00, compare destroyed Crown Towers; higher count wins.
3. If still tied, trigger rapid HP drain on remaining towers until one falls.

Extensions:

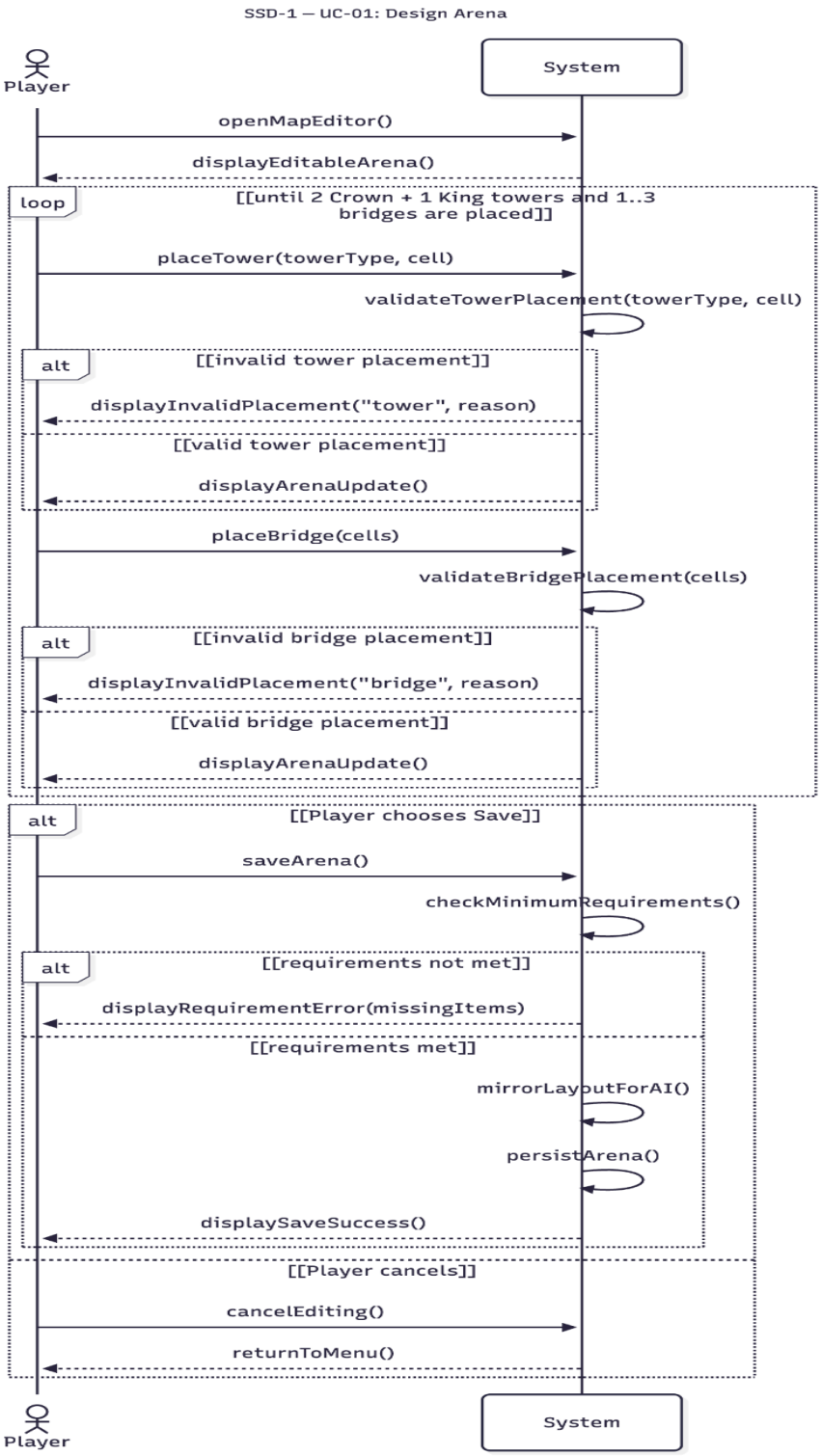
1a. **Simultaneous King Tower destruction:**

- Declare a draw.

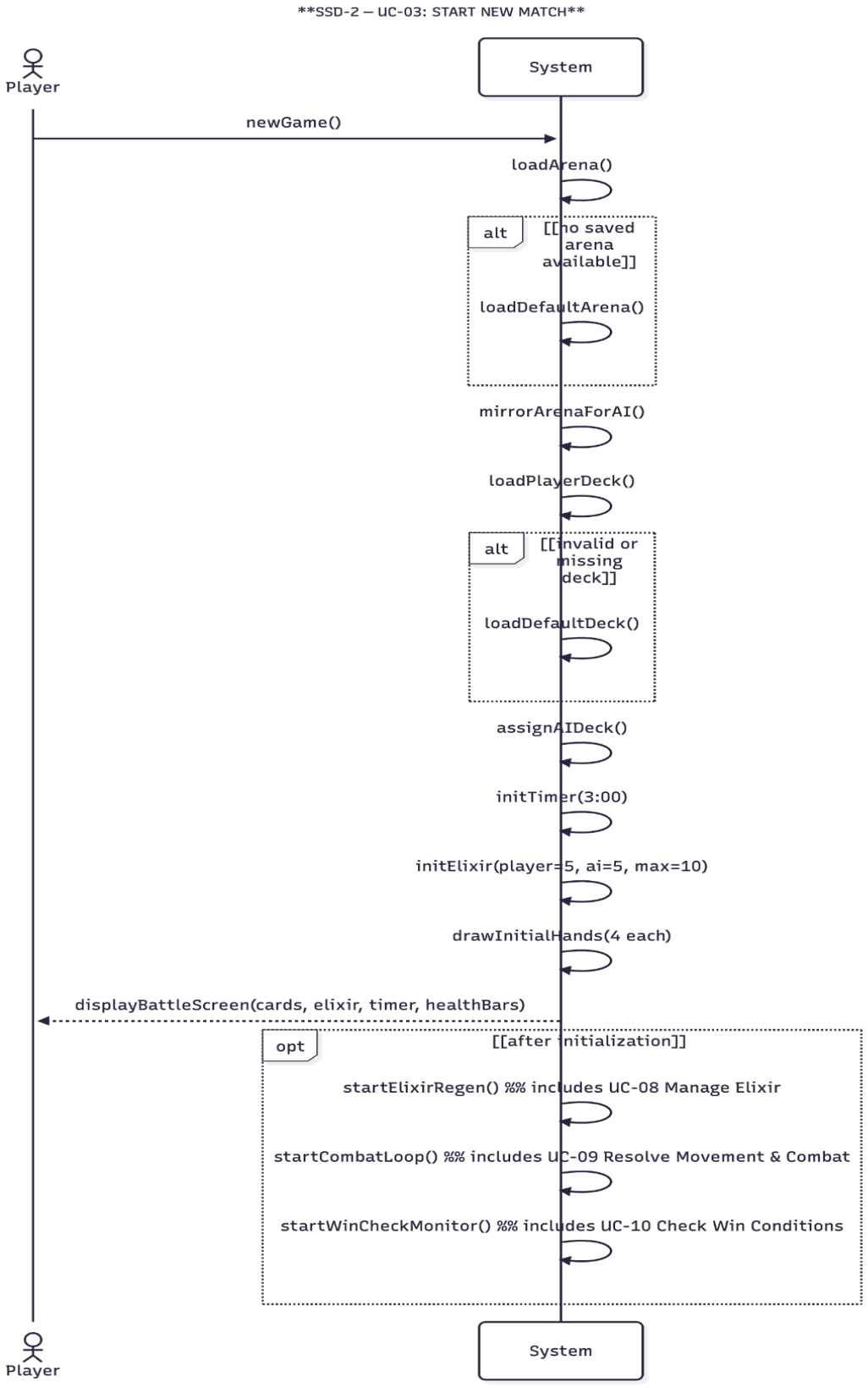
Domain Model



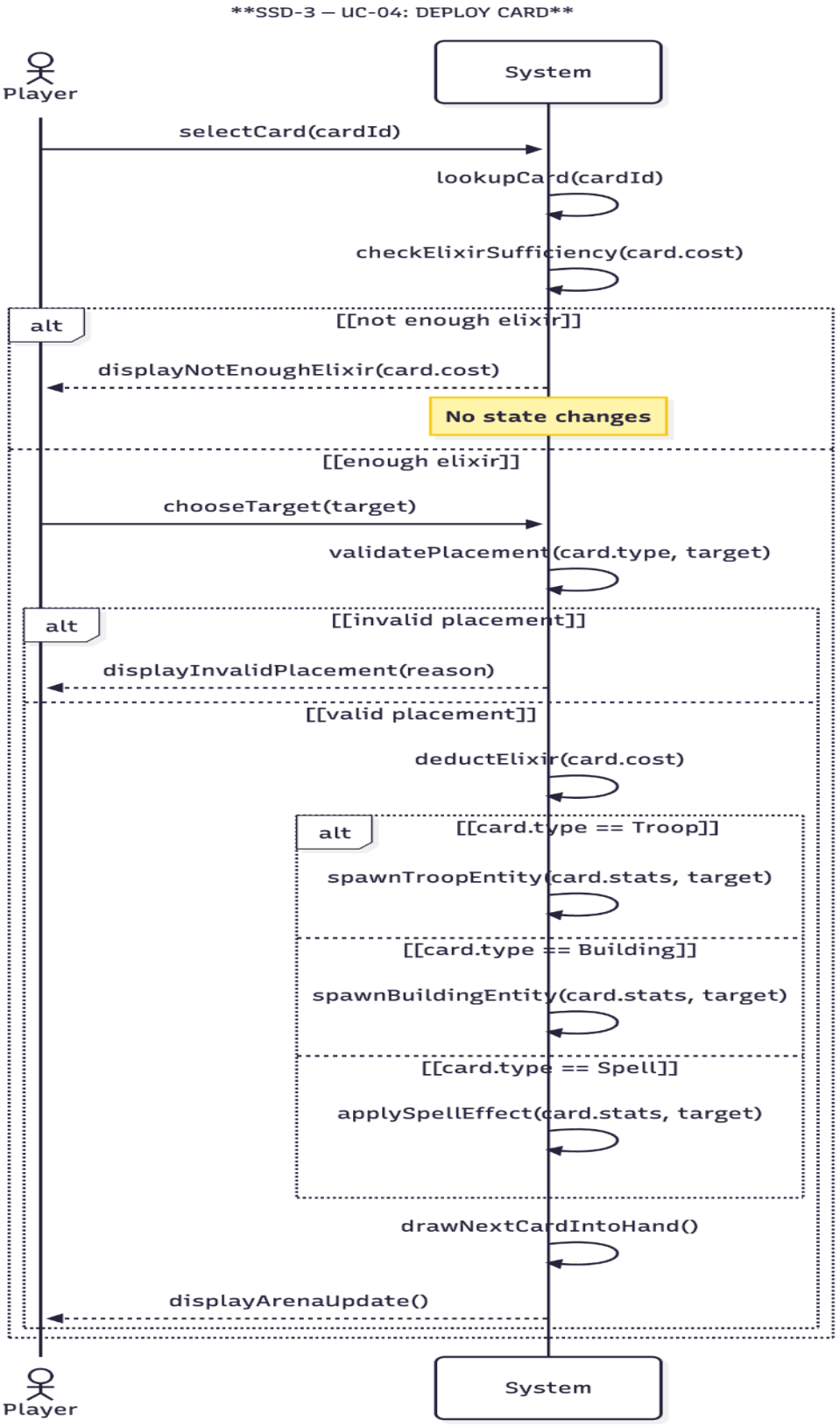
SSD 1


SSD-1 — UC-01: Design Arena
Player
System
openMapEditor()
displayEditableArena()
loop
[[until 2 Crown + 1 King towers and 1..3 bridges are placed]]
placeTower(towerType, cell)
validateTowerPlacement(towerType, cell)
alt
[[invalid tower placement]]
displayInvalidPlacement("tower", reason)
[[valid tower placement]]
displayArenaUpdate()
placeBridge(cells)
validateBridgePlacement(cells)
alt
[[invalid bridge placement]]
displayInvalidPlacement("bridge", reason)
[[valid bridge placement]]
displayArenaUpdate()
alt
[[Player chooses Save]]
saveArena()
checkMinimumRequirements()
alt
[[requirements not met]]
displayRequirementError(missingItems)
[[requirements met]]
mirrorLayoutForAI()
persistArena()
displaySaveSuccess()
[[Player cancels]]
cancelEditing()
returnToMenu()
Player
System

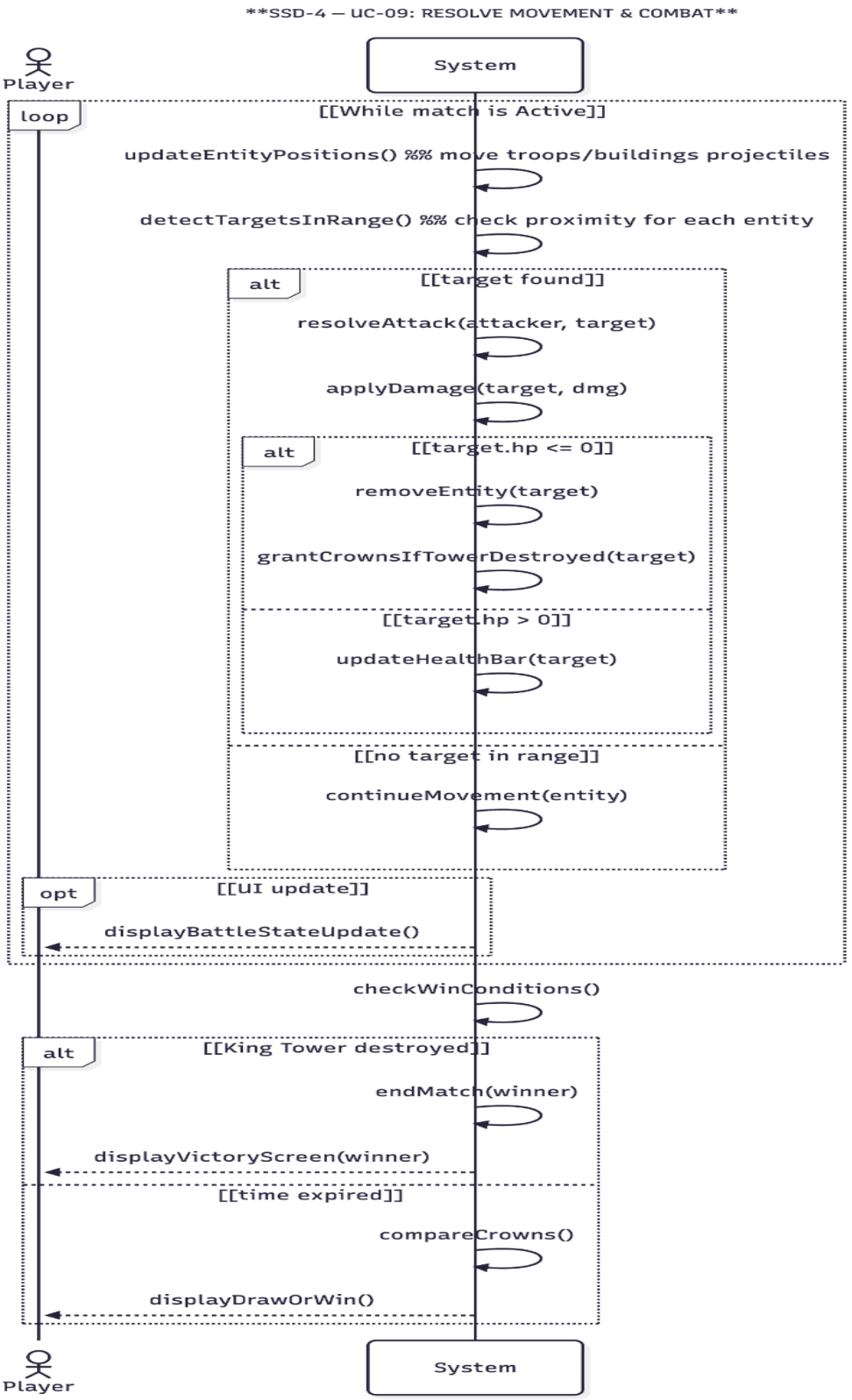
SSD 2


SSD-2 — UC-03: START NEW MATCH
Player
System
newGame()
loadArena()
alt
[[no saved arena available]]
loadDefaultArena()
mirrorArenaForAI()
loadPlayerDeck()
alt
[[invalid or missing deck]]
loadDefaultDeck()
assignAIDeck()
initTimer(3:00)
initElixir(player=5, ai=5, max=10)
drawInitialHands(4 each)
displayBattleScreen(cards, elixir, timer, healthBars)
opt
[[after initialization]]
startElixirRegen() %% includes UC-08 Manage Elixir
startCombatLoop() %% includes UC-09 Resolve Movement & Combat
startWinCheckMonitor() %% includes UC-10 Check Win Conditions
Player
System

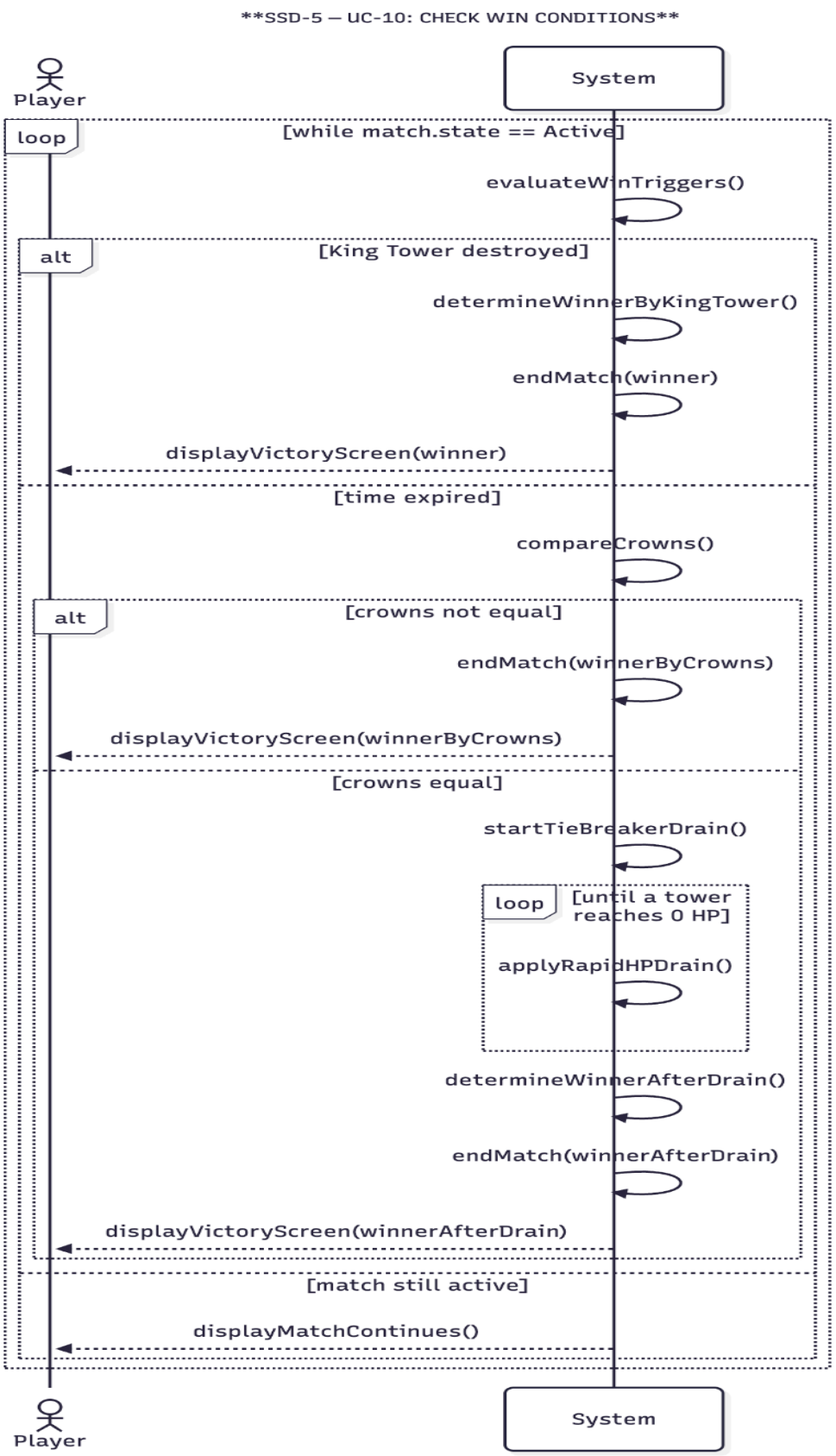
SSD 3


SSD-3 — UC-04: DEPLOY CARD
Player
System
selectCard(cardId)
lookupCard(cardId)
checkElixirSufficiency(card.cost)
alt
[[not enough elixir]]
displayNotEnoughElixir(card.cost)
No state changes
[[enough elixir]]
chooseTarget(target)
validatePlacement(card.type, target)
alt
[[invalid placement]]
displayInvalidPlacement(reason)
[[valid placement]]
deductElixir(card.cost)
alt
[[card.type == Troop]]
spawnTroopEntity(card.stats, target)
[[card.type == Building]]
spawnBuildingEntity(card.stats, target)
[[card.type == Spell]]
applySpellEffect(card.stats, target)
drawNextCardIntoHand()
displayArenaUpdate()
Player
System

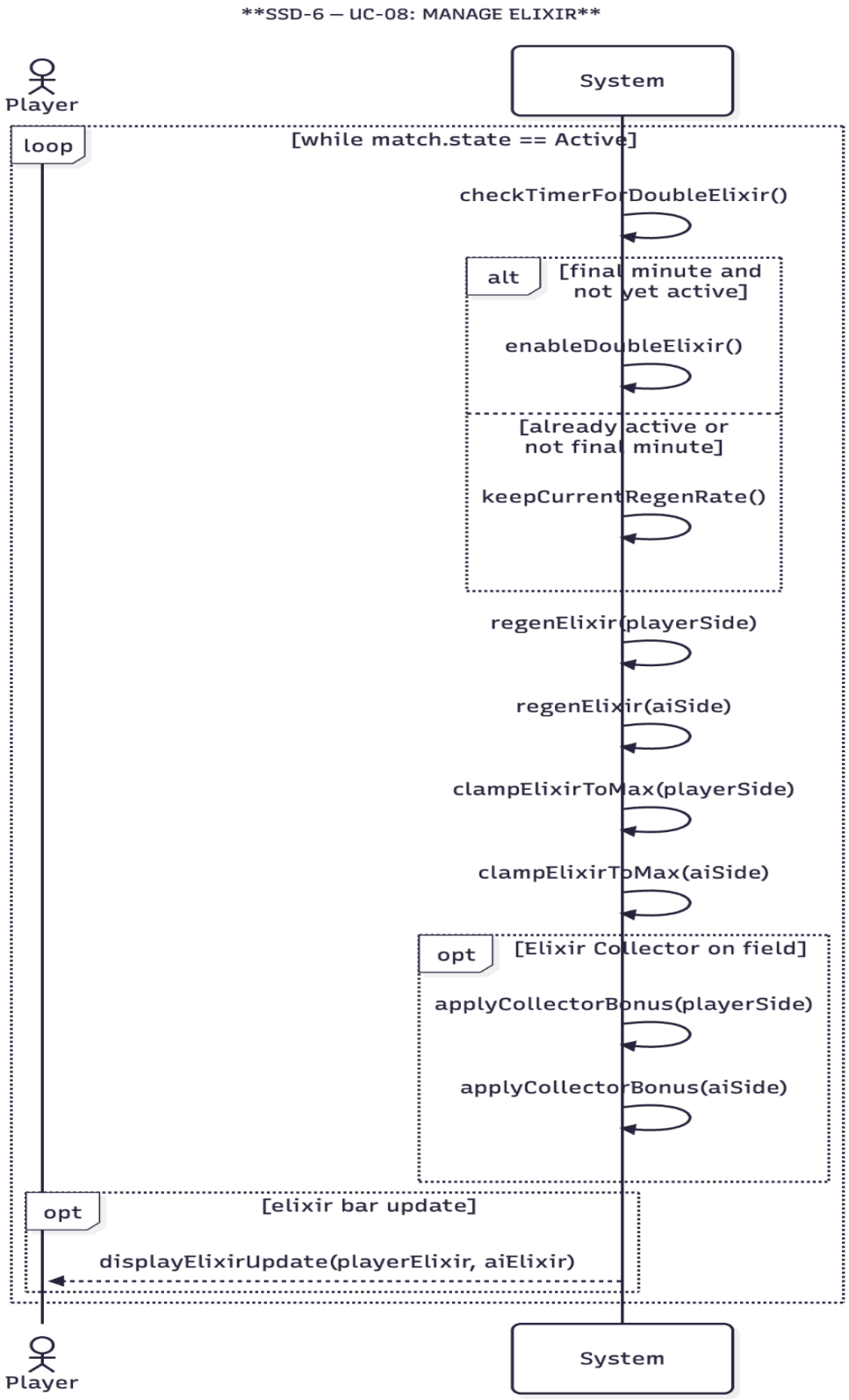
SSD 4


SSD-4 — UC-09: RESOLVE MOVEMENT & COMBAT
Player
System
loop
[[While match is Active]]
updateEntityPositions() %% move troops/buildings projectiles
detectTargetsInRange() %% check proximity for each entity
alt
[[target found]]
resolveAttack(attacker, target)
applyDamage(target, dmg)
alt
[[target.hp <= 0]]
removeEntity(target)
grantCrownsIfTowerDestroyed(target)
[[target.hp > 0]]
updateHealthBar(target)
[[no target in range]]
continueMovement(entity)
opt
[[UI update]]
displayBattleStateUpdate()
checkWinConditions()
alt
[[King Tower destroyed]]
endMatch(winner)
displayVictoryScreen(winner)
[[time expired]]
compareCrowns()
displayDrawOrWin()
Player
System

SSD 5

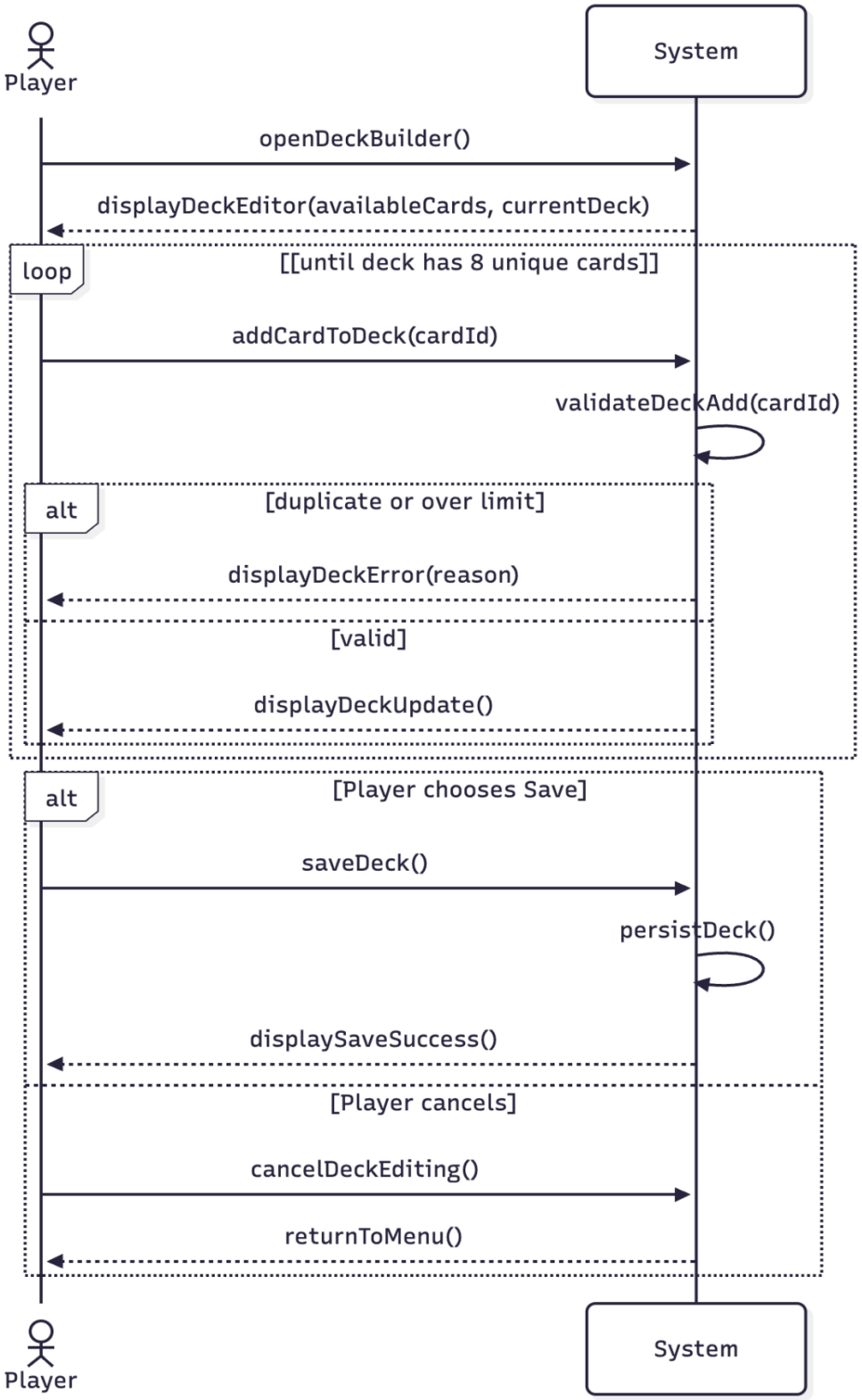

SSD-5 — UC-10: CHECK WIN CONDITIONS
Player
System
loop
[while match.state == Active]
evaluateWinTriggers()
alt
[King Tower destroyed]
determineWinnerByKingTower()
endMatch(winner)
displayVictoryScreen(winner)
[time expired]
compareCrowns()
alt
[crowns not equal]
endMatch(winnerByCrowns)
displayVictoryScreen(winnerByCrowns)
[crowns equal]
startTieBreakerDrain()
loop
[until a tower reaches 0 HP]
applyRapidHPDrain()
determineWinnerAfterDrain()
endMatch(winnerAfterDrain)
displayVictoryScreen(winnerAfterDrain)
[match still active]
displayMatchContinues()
Player
System

SSD 6

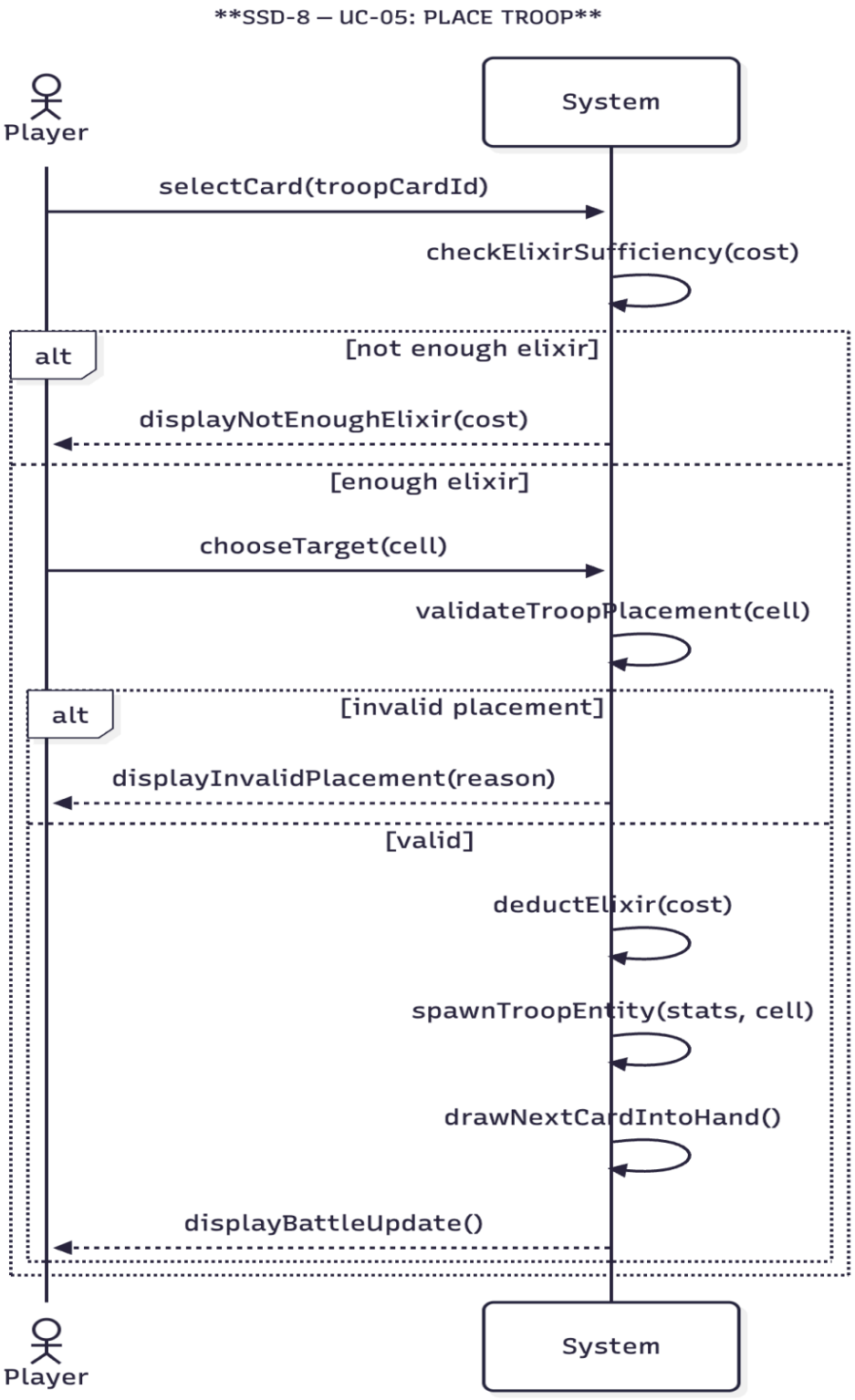

SSD-6 — UC-08: MANAGE ELIXIR
Player
System
loop
[while match.state == Active]
checkTimerForDoubleElixir()
alt
[final minute and not yet active]
enableDoubleElixir()
[already active or not final minute]
keepCurrentRegenRate()
regenElixir(playerSide)
regenElixir(aiSide)
clampElixirToMax(playerSide)
clampElixirToMax(aiSide)
opt
[Elixir Collector on field]
applyCollectorBonus(playerSide)
applyCollectorBonus(aiSide)
opt
[elixir bar update]
displayElixirUpdate(playerElixir, aiElixir)
Player
System

SSD 7

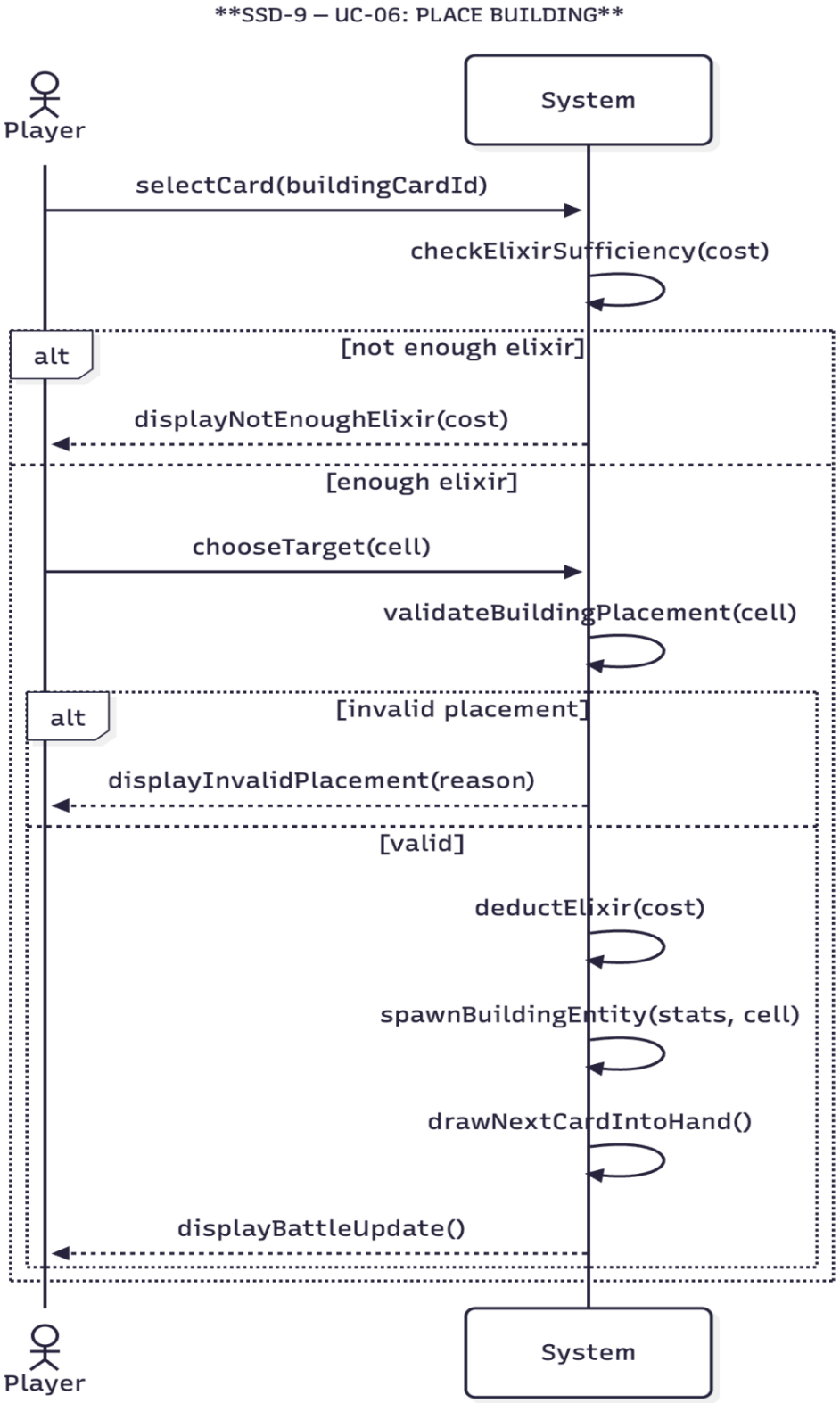
SSD-7 — UC-02: BUILD DECK



SSD 8

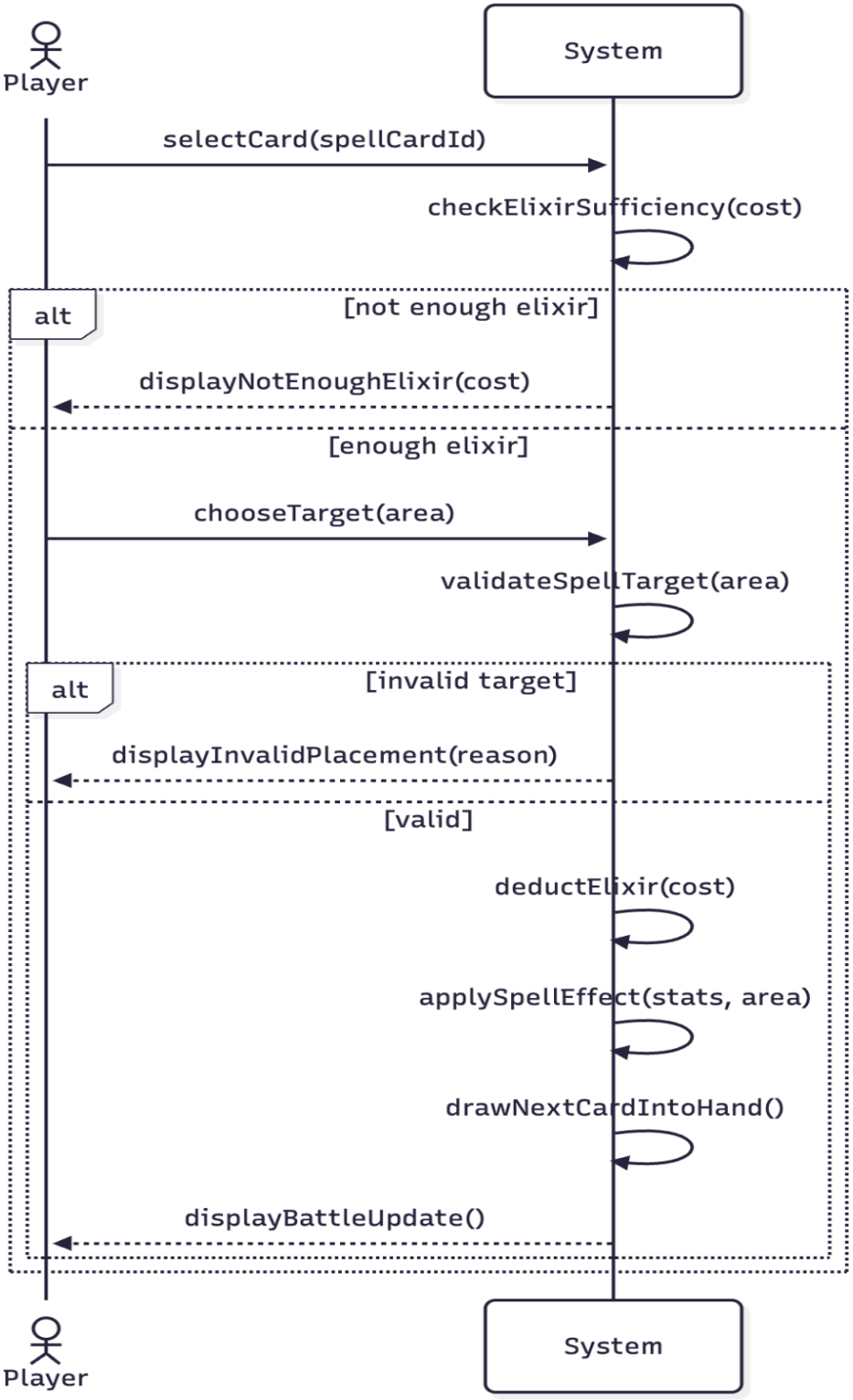

SSD-8 — UC-05: PLACE TROOP
Player
System
selectCard(troopCardId)
checkElixirSufficiency(cost)
alt
[not enough elixir]
displayNotEnoughElixir(cost)
[enough elixir]
chooseTarget(cell)
validateTroopPlacement(cell)
alt
[invalid placement]
displayInvalidPlacement(reason)
[valid]
deductElixir(cost)
spawnTroopEntity(stats, cell)
drawNextCardIntoHand()
displayBattleUpdate()
Player
System

SSD 9


SSD-9 — UC-06: PLACE BUILDING
Player
System
selectCard(buildingCardId)
checkElixirSufficiency(cost)
alt
[not enough elixir]
displayNotEnoughElixir(cost)
[enough elixir]
chooseTarget(cell)
validateBuildingPlacement(cell)
alt
[invalid placement]
displayInvalidPlacement(reason)
[valid]
deductElixir(cost)
spawnBuildingEntity(stats, cell)
drawNextCardIntoHand()
displayBattleUpdate()
Player
System

SSD 10

SSD-10 — UC-07: CAST SPELL



Operation Contracts

Operation: startNewMatch()

Cross References: UC-03 – Start New Match

Pre-conditions:

- No other match is currently active.

Post-conditions:

- The selected or default Arena was loaded.
- Player and AI decks were initialized (4 cards each).
- Elixir values were set (player = 5, AI = 5, max = 10).
- The Timer was started with 3 minutes remaining.
- The match state was set to Active and the battle screen was displayed.

Description: Initializes a new match by loading the arena, decks, and timers, then transitions the system to Battle Mode.

Operation: saveArena(layout)

Cross References: UC-01 – Design Arena

Pre-conditions:

- The player is in Arena Editor mode.
- The layout contains a valid number of towers and bridges.

Post-conditions:

- The arena layout was saved to the system.
- A mirrored AI layout was generated for the opponent side.
- A confirmation message was displayed to the player.

Description: Saves the player-designed arena and prepares the AI layout for future matches.

Operation: validateTowerPlacement(towerType, cell)

Cross References: UC-01 – Design Arena

Pre-conditions:

- The player is placing a tower on a valid grid cell.

Post-conditions:

- If valid, the tower placement was approved for display.
- If invalid, an error message was displayed with the reason (e.g., overlap or wrong side).

Description: Checks the validity of a tower placement based on position and type.

Operation: saveDeck(deck)

Cross References: UC-02 – Build Deck

Pre-conditions:

- The deck contains exactly eight unique cards.

Post-conditions:

- The deck was saved to the player profile.
- The system confirmed the successful save action.

Description: Stores the player's deck configuration for future matches.

Operation: deployCard(cardId, target)

Cross References: UC-04 – Deploy Card

Pre-conditions:

- The selected card is in the player's hand.
- The player has sufficient elixir to play the card.
- The target position is valid for the card type.

Post-conditions:

- Elixir was deducted by the card cost.
- A Troop, Building, or Spell entity was spawned according to the card type.
- The next card was drawn into the player's hand.
- The arena display was updated.

Description: Handles the entire deployment process for troop, building, or spell cards.

Operation: manageElixirTick()

Cross References: UC-08 – Manage Elixir

Pre-conditions:

- The match is active and the timer is running.

Post-conditions:

- Elixir was regenerated for both sides according to the current rate.
- If the final minute began, Double Elixir was activated.
- Elixir values were clamped to their maximum limit.

Description: Controls elixir regeneration and the activation of Double Elixir mode.

Operation: resolveMovementAndCombat()

Cross References: UC-09 – Resolve Movement and Combat

Pre-conditions:

- The match is active and entities are present on the arena.

Post-conditions:

- All entities moved according to their speed and path.
- Targets within range were attacked, and damage was applied.
- Destroyed entities were removed from the arena.
- Crown scores were updated for destroyed towers.

Description: Simulates a single combat cycle, including movement, attacks, and damage resolution.

Operation: applyDamage(attacker, target, damage)

Cross References: UC-09 – Resolve Movement and Combat

Pre-conditions:

- The attacker and target are valid entities.
- The target has remaining health.

Post-conditions:

- The target's HP was reduced by the damage value.
- If HP reached zero, the entity was removed and crowns were granted to the opponent.

Description: Applies damage to a target entity and updates battle state accordingly.

Operation: checkWinConditions()

Cross References: UC-10 – Check Win Conditions

Pre-conditions:

- The match is active.

Post-conditions:

- If a King Tower was destroyed, the match ended immediately.
- If time expired, the winner was determined by crowns or tie-breaker HP drain.
- A victory or draw screen was displayed to the player.

Description: Determines match outcome based on crowns, tower status, and remaining time.

Operation: spawnTroopEntity(stats, cell)

Cross References: UC-05 – Place Troop

Pre-conditions:

- The target cell is valid for troop placement.
- The player has enough elixir.

Post-conditions:

- A new TroopEntity was created and added to the arena.
- The entity was linked to its owning player side.

Description: Creates a troop unit on the arena and registers it for combat simulation.

Operation: spawnBuildingEntity(stats, cell)

Cross References: UC-06 – Place Building

Pre-conditions:

- The target cell is valid for a building.
- The player has enough elixir.

Post-conditions:

- A BuildingEntity was spawned and registered on the arena.
- Its lifetime and spawn timer were initialized.

Description: Creates a building structure and adds it to the combat loop.

Operation: applySpellEffect(stats, area)

Cross References: UC-07 – Cast Spell

Pre-conditions:

- The target area is within the arena bounds.
- The spell card is valid and sufficient elixir is available.

Post-conditions:

- A SpellEffect was created and applied to entities in the selected area.
- Effects such as stun, splash, or burn were processed for the given duration.

Description: Executes a spell's area-of-effect logic and updates affected entities.

Vision

KU Royale is a strategic, card-based battle simulation game inspired by tower defense mechanics.
Players construct customized arenas, assemble balanced decks, and engage in real-time matches against AI opponents.
The core objective is to destroy the enemy's towers while defending one's own through tactical deployment of troops, buildings, and spells.
The game encourages both creativity and strategy — combining spatial reasoning (in arena design) with fast-paced decision-making (during battle).
KU Royale aims to deliver an educational yet entertaining experience that emphasizes timing, resource management, and intelligent planning.

Game Modes

1. **Build Mode:**
    - Players construct arenas by arranging defensive and decorative elements on a grid.
    - Each arena type has a minimum object requirement:
        i. **Grass Arena:** At least 6 objects
        ii. **Stone Arena:** At least 9 objects
        iii. **Water Arena:** At least 13 objects
        iv. **Lava Arena:** At least 17 objects
    - Designs are saved automatically before entering Play Mode.
2. **Play Mode:**
    - Each player begins with 3 towers and a match timer of 3 minutes.
    - The goal is to destroy the opponent's King Tower while protecting their own.
    - Elixir regenerates continuously, enabling tactical deployment throughout the match.

Player Actions

- **Movement:** Units automatically traverse the arena according to their card-defined behavior.
- **Interactions:**
    - Deploy cards from the player's deck to influence battle flow.
    - Monitor elixir usage and regeneration timing.
- **Strategic Use of Cards:**
    - Time card deployment for maximum efficiency.
    - Combine troop, building, and spell effects to gain positional or defensive advantages.

Winning and Losing

- **Victory:** Opponent's King Tower destroyed or highest crown count when time expires.
- **Defeat:** Player's King Tower destroyed or lower crown count at match end.
- **Draw:** Equal crown count after time expiration and overtime.

Unique Features

1. **Card Types:**
    - **Troops:** Mobile offensive or defensive units.
    - **Buildings:**Stationary defenses or resource spawners.
    - **Spells:** Instant or timed effects impacting the battlefield.
2. **Elixir Mechanic:**
    - Elixir regenerates over time, enabling continuous play.
    - Double Elixir activates during the final minute for faster-paced gameplay.
3. **AI Opponent:**
    - Simulates adaptive behavior by analyzing player strategy.
    - Randomized card selection ensures dynamic replayability.
4. **User Interface:**
    - Main menu provides access to arena editing, deck management, and match initiation.
    - In-match display shows timers, elixir bars, and crowns.
    - Pause, help, and exit buttons available during gameplay.

Non-Functional Requirements

Performance

- Game updates (e.g., movement, animations, combat logic) are processed smoothly, with response times typically below 120 milliseconds per frame.

Security

- All player data — including decks, preferences, and match progress — is stored securely and isolated from other sessions.

Usability

- The interface is designed for clarity and accessibility, supporting both novice and experienced players.
- Built-in help and tooltips explain mechanics directly within the game.

Extensibility

- The system architecture supports seamless addition of new arenas, cards, and AI types without altering existing components.

Scalability

- The interface and performance adapt efficiently across devices and resolutions, maintaining consistent gameplay quality..

GLOSSARY

Player: Human user who controls gameplay actions and manages decks.
Arena: The playfield grid where battles take place.
Deck: The set of 8 cards the player selects for a match.
Elixir: Regenerating resource used to deploy cards.
Troop / Building / Spell: Categories of cards with distinct behavior types.
Crown: Symbol representing tower destruction progress.
King Tower: Central structure; its destruction ends the game.
Double Elixir: Late-game period doubling elixir regeneration.
Match Timer: Countdown controlling total match duration.
AI Opponent: Computer-controlled enemy that adapts to player tactics.

AI and External Tool Usage

We used OpenAI's ChatGPT (GPT-5 model) as a generative AI assistant during the preparation of this report.
The AI was used only for writing support, grammar refinement, and ensuring consistency between use case narratives, SSDs, operation contracts, and the domain model.
All technical content, logic, and UML structures were created and validated by our team.

We also used MermaidChart (https://www.mermaidchart.com) to design and visualize UML diagrams, including use case diagrams, domain models, and system sequence diagrams.

Type of Queries

- **Clarifying Larman-style UML standards and naming conventions.**
- **Refining and rephrasing use case explanations and operation contracts for clarity.**
- **Ensuring alignment between diagrams and textual documentation.**
- **Preparing short, oral-style summaries for meeting presentations.**

Impact on the Final Work

AI output was used as a writing and editing aid only.
Every section generated with assistance was reviewed, verified, and modified by the team to reflect our own design and understanding.
MermaidChart was used exclusively as a diagram visualization tool, not as a generative model.

Citations

- **OpenAI.** ***ChatGPT (GPT-5).*** **https://chat.openai.com**
- **MermaidChart.** ***UML and Flowchart Visualization Tool.*** **https://www.mermaidchart.com**