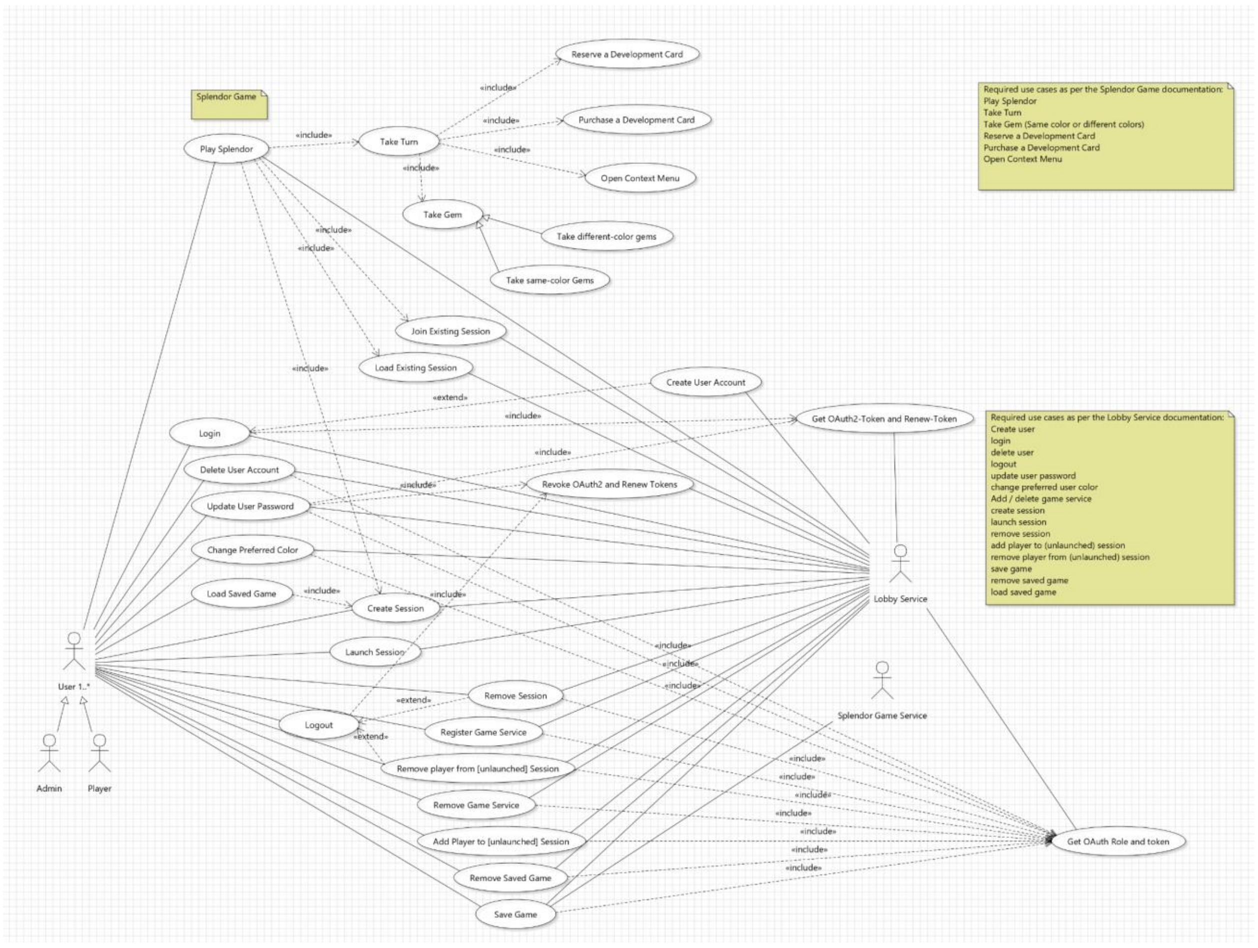
Splendor Use Case Diagram:

Berardelli, Lawrence
Fiore, Sofia
Hayden, Zach
Krishnan, Ojas
Léon, Jeff
Sklokin, Svyatoslav



Required use cases as per the Splendor Game documentation:
Play Splendor
Take Turn
Take Gem (Same color or different colors)
Reserve a Development Card
Purchase a Development Card
Open Context Menu

Required use cases as per the Lobby Service documentation:
Create user
login
delete user
logout
update user password
change preferred user color
Add / delete game service
create session
launch session
remove session
add player to (unlaunched) session
remove player from (unlaunched) session
save game
remove saved game
load saved game

PlaySplendor

Use Case: PlaySplendor
Scope: Splendor
Level: User Goal
Intention in Context: The intention of the *Player* is to play a game of Splendor against other players.
Multiplicity: Multiple *Players* can play the game simultaneously, however only one *Player* may play a turn at a time. A *Player* is also not allowed to play multiple game sessions simultaneously.
Primary Actor: *Player*
Secondary Actors: *Player* (who play the role of opponents)
Main Success Scenarios:

1. *Player* logs onto *System*.
2. *System* displays the available game sessions to the *Player*.
3. The *Player* has the option to <u>create a new session</u>, <u>join an existing session</u> or <u>load an existing session</u>.
4. Once enough players have joined the game, the game interface is initialised with the board, card and token piles, and *Player* inventories.
5. *Players* <u>take turns</u>.
6. *System* informs *Players* of the winner of the game.

Extensions:
4a. *Player* was unable to create, join or load a session, in which case the use case continues at step 3.

TakeTurn

Use Case: TakeTurn
Scope: Splendor
Level: Sub Function
Intention in Context: Intention of the Players is to take their turn.
Multiplicity: Only one Player can take their turn simultaneously.
Primary Actor: Player
Secondary Actors: Player (who play the role of opponent)
Main Success Scenario:
1. *System* informs current *Player* that it is their turn
2. Current *Player* informs *System* that they would like to perform one of the following actions:

- <u>taking two gems of the same colour</u>
- <u>taking three gems of different colours</u>
- <u>reserving a development card</u>
- <u>purchasing a development card</u>

3. Current *Player* informs *System* that they would like to end their turn.
4. *System* informs the current *Player* whether they qualify for a noble card
Extensions:
3a. *Player* informs *System* that they would like to undo the last action; use case continues at step 2.
4a. *System* determines that current *Player* qualifies for multiple noble cards and informs the current *Player* that they must choose a noble card
    4a.1 Current *Player* informs *System* which noble card they would like to choose; use case ends in success

TakeSameColorGems

Use Case: TakeSameColouredGems
Scope: Splendor
Level: Sub Function
Intention in Context: Intention of the Players is to take 2 gem tokens of the same colour from the gem piles and add them to their inventory.
Multiplicity: Only one Player can take gem tokens simultaneously.
Primary Actor: Player
Secondary Actors: Player (who play the role of opponent)
Main Success Scenario:
1. Current *Player* informs *System* that they would like to take two gems of the same colour.
2. *System* informs current *Player* which piles they can choose from (the pile cannot have less than 4 tokens in it).
3. Current *Player* informs *System* which pile they would like to choose 2 gems from.
4. *System* informs *Players* of new game state.

Extensions:
2a. *System* determines that there are no possible piles for the current *Player* to choose; the use case ends in failure
3a. *System* determines that the amount of gem tokens in the current *Player*'s inventory exceeds 10
    3a.1. *System* informs *Player* to return tokens until they only have 10 tokens left in their inventory.
    3a.2. *Player* informs *System* which tokens they would like to return; the use case continues at step 4.

TakeDifferentColouredGems

Use Case: TakeDifferentColouredGems
Scope: Splendor
Level: Sub Function
Intention in Context: Intention of the Players is to take 3 gem tokens of different colours from the gem piles and add them to their inventory.
Multiplicity: Only one Player can take gem tokens simultaneously.
Primary Actor: Player
Secondary Actors: Player (who can view the current game state)
Main Success Scenario:
1. Current *Player* informs *System* that they would like to take three gems of the same colour.
2. *System* informs current *Player* which piles they can choose from (the pile cannot be empty).
3. Current *Player* informs *System* which gem token they would like to take.
Steps 2 and 3 are repeated for each token.
4. *System* informs *Players* of new game state.

Extensions:
2a. *System* determines that there are no possible piles for the *Player* to choose; the use case ends in failure
3a. *System* determines that the amount of gem tokens in the current *Player*'s inventory exceeds 10.

3a.1. *System* informs current *Player* to return tokens until they only have 10 tokens left in their inventory.

3a.2. Current *Player* informs *System* which tokens they would like to return; the use case continues at step 4.

PurchaseDevelopmentCard

Use Case: PurchaseDevelopmentCard
Scope: Splendor
Level: Sub Function
Intention in Context: Intention of the Player is to purchase a development card of their choice from the game board by spending gem tokens as necessitated by the cost detailed on the card and adding it to their inventory.
Primary Actor: Player
Secondary Actors: Player (who play the role of opponents)
Main Success Scenarios:

1. Current *Player* informs *System* that they would like to purchase a development card of their choice from the game board.
2. *System* adds the card to the current *Player*'s inventory, removes the required tokens from their inventory and informs *Players* of the new game state.
3. *System* replaces the purchased card with a new card from the deck and informs *Players* of the new game state.

Extensions:
2a. *System* determines that the token cost values of the development card does not match the numbers of the tokens and bonuses in the player's inventory; the use case ends in failure.
2b. (Orient Expansion) The purchased development card allows the current *Player* to claim a free card of a particular level,

2b.1. *System* prompts the current *Player* to select a free card from that level of the game board.

2b.2. *Player* informs *System* which card they have selected.

2b.3. *System* adds that card to the *Player*'s inventory, use case continues at step 3.

2c. (Orient Expansion) The purchased development card allows a Player to pair it with another development card to increment the gem bonus

2c.1. *System* prompts the current *Player* to select a card from their inventory for pairing.

2c.2. Current *Player* informs the *System* which card they have selected.

2c.3. *System* increments the bonus counter of the gem corresponding to the *Player*'s choice and inform *Players* of the new game state; use case continues at step 3.

3a. *System* determines that the number of cards in the deck pile of the corresponding row has been reduced to 0

3a.1. *System* does not replace the card slot in the with a new development card; use case ends in success.

ReserveDevelopmentCard

Use Case: ReserveDevelopmentCard

Scope: Splendor

Level: Subfunction

Intention in Context: The intention of the *Players* is to reserve a chosen development card as well as get one gold token if available.

Primary Actor: *Player*

Secondary Actors: *Player (who play the role of opponent)*

Main Success Scenario:

1. Current *Player* informs *System* about which development card from the game board (including Orient cards and face-down decks) they would like to reserve.
2. *System* replenishes card if there are still cards of corresponding category left and informs *Players* of new game state.

Extensions:

1a. *System* determines that the current *Player* already has three cards reserved. The use case ends in failure.

1b. *System* determines that the number of tokens in the current *Player*'s inventory exceeds 10.

- 1b.1. *System* informs *Player* to return tokens until *Player* only has 10 or less tokens left in their inventory.
- 1b.2. *Player* informs *System* which tokens *Player* would like to return. The use case continues at step 2.

OpenContextMenu

Use Case: OpenContextMenu

Scope: Splendor

Level: Sub Function

Intention in Context: Intention of the Player is to use the context menu of a card in order to perform an action other than the default purchase of the card; reserving the card or viewing its respective tooltip.

Primary Actor: Player

Secondary Actor: Player (who can view and interact with the context menu triggered on a card)

Main Success Scenarios:

1. Current *Player* triggers a context menu on a card of their choice.
2. *System* informs the current *Player* that they can either reserve a card from the game board or view the tooltip for that particular card.
3. *Player* informs *System* of their choice.

Extensions:

3a. Current *Player* informs *System* that they would like to view the tooltip for the chosen card.

- 3a.1 The *System* displays the tooltip information for the card to the *Player*.

Save Game

Use Case: Save Game
Scope: Lobby Service UI
Level: User Goal
Intention in Context: The user intends to save the data pertaining to their current game session in a registered game service.
Multiplicity: Only the admin who registered the game-service may register a save game.
Primary Actor: User
Secondary Actor: Lobby Service
Main Success Scenario:

The *User* pushes the save game button.
The save game button sends an authentication request to the *Lobby Service* to ensure that who pressed the button is the admin that registered the game-service.
Lobby Service registers a saved game.

Extensions:

2a. The authentication fails and the game is not saved.

Remove Save Game

Use Case: Remove Save Game
Scope: Lobby Service UI
Level: User Goal
Intention in Context: The user intends to remove the saved data pertaining to a previous game session in a registered game service while also implicitly removing all unlaunched sessions forked from the saved game in question.
Multiplicity: Only the admin who registered the game-service may remove a previously saved game.
Primary Actor: User
Secondary Actor: Lobby Service
Main Success Scenario:

The *User* selects the previous save game and pushes the remove save game button.
The remove save game button sends an authentication request to the *Lobby Service* to ensure that who pressed the button is the admin that registered the game-service.
Lobby Service removes the saved game and implicitly removes all unlaunched sessions forked from this now removed save game.

Extensions:

2a. The authentication fails and the save game is maintained.

Register Game Service

Use Case: Register Game Service
Scope: *Lobby Service Client*
Level: User Goal
Intention in Context: Intention of the user is to register a game service.
Multiplicity: Multiple *users* can register game services simultaneously.
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:
User requests *Lobby Service Client* to register a game service

1. *User* requests *Lobby Service Client* to register a game service
2. *Lobby Service Client* sends the request to the *System*
3. *System* informs *User* that the game service registration was sucessful

Remove Game Service

Use Case: Remove Game Service
Scope: *Lobby Service Client*
Level: User Goal
Intention in Context: Intention of the user is to remove a game service
Multiplicity: Multiple users can remove game services simultaneously.
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:

1. *User* requests *Lobby Service Client* to register a game service
2. *Lobby Service Client* sends the request to the *System*
3. *System* successfully removes the game service and informs the *Use*

Create Session

Use Case: Create Session
Scope: Lobby Service
Level: User Goal
Intention in Context: Intention of the user is to create a session which contains the game's name, the creator's name and the number of players (i.e. current number of players/maximum).
Multiplicity: Multiple users can create sessions simultaneously.
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:

1. *User* requests *System* to create a session
2. *System* sends the request to the *Lobby Service Client*
3. *Lobby Service Client* accepts request and informs the *System*
4. *System* successfully creates a session and informs the *User*

Launch Session

Use Case: Launch Session
Scope: Splendor Game System
Level: User Goal
Intention in Context: Intention of the user is to launch a session
Multiplicity: Multiple users can launch sessions simultaneously.
Primary Actor: User
Secondary Actors: Splendor Game System, Lobby Service
Main Success Scenario:

1. *User* requests *System* to launch a session
2. *System* sends the request to the *Lobby Service Client*
3. *Lobby Service Client* accepts request and informs the *System*
4. *System* successfully launches the session and informs the *User*

Extensions:
3a. If the number of registered players is invalid, *Lobby Service Client* declines request and informs the *system*
4a. System does not react to the user's request if the number of registered players in the session is invalid.

Remove Session

Use Case: Remove Session
Scope: Splendor Game System
Level: User Goal
Intention in Context: Intention of the user is to remove a session
Multiplicity: Multiple users can remove sessions simultaneously
Primary Actor: User
Secondary Actors: Splendor Game System, Lobby Service
Main Success Scenario

1. *User* requests *System* to remove a session
2. *System* sends the request to the *Lobby Service Client*
3. *Lobby Service Client* accepts request and informs the *System*
4. *System* informs *User* that the session has successfully been created

Add Player to [unlaunched] Session

Use Case: Add Player to [unlaunched] session
Scope: Lobby Service
Level: User Goal
Intention in Context: Intention of the user is to add a player to an unlaunched session
Multiplicity: Only one user (the administrator) can add a player to an unlaunched session
Primary Actor: User (admin)

Secondary Actors: Lobby Service
Main Success Scenario:

1. *User* requests *System* to add a player to the chosen session
2. *System* sends the request to the *Lobby Service Client*
3. *Lobby Service Client* accepts request and informs the *System*
4. *System* successfully adds a player to the session and informs the *User*
5.

Extensions:
3a. The *user* has already been added and the *Lobby Service Client* declines the request
4a. *System* does not react to the *user's* request

Remove Player from [unlaunched] Session

Use Case: Remove Player from [unlaunched] Session
Scope: Lobby Service
Level: User Goal
Intention in Context: Intention of the user is to remove a player from an unlaunched session
Multiplicity: Only one user (admin) can remove a player from an unlaunched session
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:

1. *User* requests *System* to remove a player from the chosen session
2. *System* sends the request to the *Lobby Service Client*
3. *Lobby Service Client* accepts request and informs the *System*
4. *System* successfully removes a player to the session and informs the *User*

Extensions:
3a. The *Lobby Service Client* declines the request if the *user* and the player attempted to be removed are the same
4a. *System* does not react to the *user's* request

Retrieve All Users

Use Case: Retrieve All Users
Scope: Lobby Service Client
Level: User Goal
Intention in Context: Intention of admin is to view existing profiles in the lobby service
Multiplicity: Multiple admins can view this information at the same time.
Primary Actor: Admin
Secondary Actors: Lobby Service
Main Success Scenario:

1. ***User*** informs ***System*** of desire to retrieve active profiles.
2. ***System*** requests ***User***'s role to ***Lobby Service***
Only occurs if the requesting user is admin.
3. ***System*** requests data from ***Lobby Service*** and displays it to the user.
Extensions:
2.a. ***Lobby Service*** informs ***System*** that ***User*** is non-admin. Use-case ends in failure.

Get User Details

Use Case: Get User Details
Scope: Lobby Service Client
Level: Subfunction
Intention in Context: Intention of user is to retrieve the profile details of a user
Multiplicity: Multiple users can request the information of multiple users at a time
Primary Actor: User
Secondary Actors: Lobby Service

Main Success Scenario:
1. ***User*** informs ***System*** of desire to view user details
2. ***System*** requests ***User***'s role to ***Lobby Service***
3. ***System*** forwards this request to ***Lobby Service*** and displays the information to the ***User***
Extensions:
This can occur if the requesting user is non-admin and requests data for other users, among others.
3.a. ***Lobby Service*** fails to validate the request, ***System*** informs ***User,*** use case ends in failure.

Create User Account

Use Case: Create User Account
Scope: Lobby Service Client
Level: User Goal
Intention in Context: Intention of user is to create a new user profile
Multiplicity: Multiple users can create one account at a time.
Primary Actor: User
Secondary Actor: Lobby Service

Main Success Scenario:

1. ***User*** provides to ***System*** the requisite data to create a new profil
2. ***System* requests *User*'s role** to ***Lobby Service*** Only occurs if user is admin
3. ***System*** forwards this data and request to ***Lobby Service*** Extensions:

2.a. ***Lobby Service*** informs ***System*** that ***User*** is non-admin. Use-case ends in failure.
3.a. ***Lobby Service*** fails to validate the request (e.g non-compliant password), ***System*** informs ***User,*** use case ends in failure

Delete User Account

Use Case: Delete User Account
Scope: Lobby Service Client
Level: User Goal
Intention in Context: Intention of user is to delete a certain user account
Multiplicity: Multiple users can delete one account at a time
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:

1. ***User*** provides to ***System*** required data to delete a user
2. ***System* requests *User*'s role** to ***Lobby Service*** Only occurs if user is admin
3. ***System*** forwards this request to ***Lobby Service***.

Update User Password

Use Case: Update User Password
Scope: Lobby Service Client
Level: User Goal
Intention in Context: Intention of user is to update a user password
Multiplicity: Multiple users can update one password at a time.
Primary Actor: User
Secondary Actors: Lobby Service

Main Success Scenario:

1. ***User*** provides to the System the requisite data to update the password and makes the request.
2. ***System*** forwards the data and the request to ***Lobby Service*** and responds to the User indicating success.

Extensions:

1a. If ***User*** is not admin, ***User*** must provide to ***System*** extra data, indicating the current password.
2a. If ***Lobby Service*** fails to validate this request (e.g. non-compliant password) the ***System*** displays failure to the User***.*** The use case ends in failure.

Change Preferred Colour

Use Case: Change Preferred Colour
Scope: Lobby Service Client
Level: User Goal
Intention in Context: User is to change a target user's preferred colour.
Multiplicity: Many users can change one preferred colour at a time.
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:

1. ***User*** provides to ***System*** required data to specify new colour, and requests the change.
2. ***System*** forwards this request to the ***Lobby Service.***

Get OAuth Role

Use Case: Get OAuth Role
Scope: Lobby Service Client
Level: Subfunction
Intention in Context: Intention of Lobby Service is to retrieve and send User Role data to System.
Multiplicity: One Lobby Service can handle multiple role requests at a time.
Primary Actor: Lobby Service
Main Success Scenario:

1. ***System*** forwards role request data and requests role from ***Lobby Service***
2. ***Lobby Service*** responds to the ***System*** with the role of the requested user.

Get OAuth2 Token and Renew Token

Use Case: Get OAuth2 Token and Renew Token
Scope: Lobby Service Client
Level: Subfunction
Intention in Context: Intention of ***Lobby Service*** is to generate a new oauth token and send it to the ***System.***
Multiplicity: One ***Lobby Service*** can handle multiple token requests at a time.
Primary Actor: *Lobby Service*
Main Success Scenario:

1. ***System*** forwards token request data and request to ***Lobby Service***

2. ***Lobby Service*** responds to the ***System*** by returning the token pair for the target user.
Extensions
2.a. If ***Lobby Service*** is unable to identify a user by the request data it informs ***System*** and use-case ends in failure.

Revoke OAuth2 Token and Renew Token

Use Case: Revoke OAuth2 Token and Renew Token
Scope: Lobby Service Client
Level: Subfunction
Intention in Context: Intention of ***Lobby Service*** is to revoke a users oauth tokens.
Multiplicity: One ***Lobby Service*** can revoke many tokens simultaneously.
Primary Actor: *Lobby Service*
Main Success Scenario:
1. ***System*** informs ***Lobby Service*** of delete token and forwards required data
2. ***Lobby Service*** responds to the System indicating success.

Log in

Use Case: Log in
Scope: Lobby Service Client
Level: User Goal
Intention in Context: Intention of User is to identify themselves
Multiplicity: Multiple users can log in simultaneously.
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:

1. ***User*** provides log in data and request to the System***.***

2. ***System*** gets oauth2 token and renew token for provided data and responds to user indicating success by changing to matchmaking screen. Extensions:

2.a. If get token fails, ***Lobby Service*** informs ***User*** and the use case continues at step 1.

Log Out

Use Case: Log Out
Scope: Lobby Service Client
Level: User Goal
Intention in Context: Intention of User is to log out.
Multiplicity: Multiple users can log out simultaneously.
Primary Actor: User
Secondary Actors: Lobby Service
Main Success Scenario:
1. ***User*** informs ***System*** of desire to log out.
2. ***System*** revokes oauth2 token and responds to user by changing to log in screen