# Operation Model

**Contents**

# 1. Game Setup

**Operation:** LegendsOfAndor::login(username: String, password: String)
**Scope:** Player, Database
**Messages:** Player::{promptLoginResponse; promptCreateNewGame; promptOpenSavedGame; promptJoinGame;}
**Post:** The login operation looks for any existing instance of Player in the Database with the inputted username and password. If the inputted information can't be found in the Database, login will fail and System will inform player of failure. If the inputted information matches the Database, an instance of player is retrieved from the Database and promptloginSuccess.

**Operation:** LegendsOfAndor::logout()
**Scope:** Player, Database
**New:** N/A
**Messages:** Player :: {logoutSuccess;}
**Post:** This operation logs out the instance player. The database would switch the player status to offline.

**Operation:** LegendsOfAndor::createAccount(username: String, password: String, confirmPassword: String)
**Scope:** Player, Database
**New:** newAccount: Account
**Messages:** Player::{createAccountResponse;}
**Pre:** valid username, valid password
**Post:** If a valid username and password (not already existing in the database) are entered by players into the system, System will store a new account in Database with entered credentials. In case of invalid username and/or password combination, the user will not move passed login (i.e. will not be redirected to game lobby)

# 2. Game Lobby Process

**Operation:** LegendsOfAndor::replyCreateNewGame()
**Scope:** Player, Database
**New:** newGame: Game;
**Messages:** Player :: {prompSetGame;}

**Post:** The replyCreateNewGame operation creates a new game instance and saves it to the Database. The player's status becomes host. This operation would lead the host to the SetGame operation.

**Operation:** LegendsOfAndor::replySetGame(gameName: String, public: boolean, difficulty: int, numPlayer: int)
**Scope:** Player, Game, Database
**New**: N/A
**Messages**: Player :: {promptInvitePlayer; promptChooseHero;}
**Pre:** player(status)==Host
**Post:** This operation allows a host to set up the fields of a new game. The host may name a game, set the game as public or private, choose number of players, and the difficulty that the host wants to play in this game instance. The host can choose hero after creating the new game.

**Operation:** LegendsOfAndor::invitePlayer(self : Player,  username : String)
**Scope:** Game, Player, GameLobby
**New:** newAccess: Game
**Messages:** Player :: {promptInvitationResponse;}
**Pre:** player.status == online
**Post:** The invitePlayer operation allows a host to invite anyone online to join an existing game instance. There is a precondition that the invited player has to be online in the game lobby. An invitation message will be prompted to both players. This invitation operation provides access to join both private and public games.

**Operation:** LegendsOfAndor::replyInvitation(join: boolean)
**Scope:** Player, Database
**New**: N/A
**Messages:** Player :: {promptChooseHero;}
**Pre:** invitation.exists(Database) == true
**Post:** The Player chooses to accept or decline the invitation. If he accepts, an association is created between the player and the game. If he declines, back to game lobby.

**Operation:** LegendsOfAndor::replyJoinGame()
**Scope:** Player, Database
**Messages:** Player :: {promptJoinGameSession;}
**Post:** The Player selects he wants to join a game. The system shows available games that can be joined.

**Operation:** LegendsOfAndor:: replyChooseHero(h: Hero)
**Scope:** Player, Hero
**New:** newHero: Hero
**Messages:** Player :: {promptDistributeWineskinAndGold;
promptHeroTaken_e;}
**Pre:** hero must not be taken by another player
**Post:** The replyChooseHero operation allows all players to select their hero. If a player wants a hero that is selected already, a heroTaken message will be displayed to the player. If a player successfully selects hero, a hero selected message will be prompted.

**Operation:** LegendsOfAndor:: replyDistributeWineskinAndGold
(amountOfGold : int, amountOfWineskin : int)
**Scope:** Player, Articles
**New:** newArticle: Article
**Messages:** Player :: {promptPlayerOrder;}
**Post:** The replyDistributeWineskinAndGold operation happens after all heroes are set. All players are allowed to *communicate* to discuss how they would want to distribute the gold and wineskin given in the beginning of the game.

**Operation:** LegendsOfAndor::replyOpenSavedGame()
**Scope:** Player, Database
**Messages:** Player :: {promptSavedGames;}
**Post:** The Players selects the saved game they would like to open. The system shows available saved games for the player to select. This operation would lead the players to the waiting lobby to load the saved game.

**Operation:** LegendsOfAndor::replyLoadSavedGame(filePath: String)
**Scope:** Player, Database

**Messages:** Player :: {msgWaitForPlayers;}
**Pre:** filePath.exists(Database) == true
**Post:** The Player selects which game they would like to load. The system receives the game location from filePath which will create an association between the player and the game and will output a “Waiting for other players” message. When all players are ready online, the saved game would start loading.

**Operation:** LegendsOfAndor::replyJoinGameSession(gameID: String)
**Scope:** Player, Database
**New**: N/A
**Messages:** Player :: {promptJoin; or promptSessionFull_e;}
**Pre:** gamePhase == public
**Post:** The Player selects which public game they would like to join. The system receives the name of the game which will be associated to the host game. If the game is full, a promptSessionFull exception will be prompted.

**Operation:** LegendsOfAndor::leaveLobby()
**Scope:** Game, Player, GameLobby
**New:** N/A
**Messages:** Player :: {msgPlayerLeft;}
**Pre:** Player must be in an existing game
**Post:** The leave operation allows a player to leave an existing game in the game lobby. All other players in the game will be notified with a player left message if this operation is triggered. This operation will allow new players to join the current existing game.

**Operation:** LegendsOfAndor::replyPlayerOrder(gamePlayerList: list)
**Scope:** Player
**New:** newGamePlayList: Order
**Messages:** Player :: {promptStartGame;}
**Post:** The playerOrder operation allows all players to place themselves to set the starting order. The operation will lead all players to the sunrise box and start playing. This is the last step of the pre-game process.

**Operation:** LegendsOfAndor::readyPlayer(player_ready: boolean)

**Scope:** Game, Player, GameLobby
**Messages:** Player :: {promptReadyPlayer;}
**Pre:** The player is not ready.
**Post:** The player is tagged as ready.

**Operation:** LegendsOfAndor::startGame()
**Scope:** Map, Player, Creature, Tokens, Card
**Messages:** Player :: {promptTakeTurn;}
**Pre:** non-host players are all ready.
**Post:** The playGame operation allows host to start a game and then leads all players to the main game board and preset all tokens including merchants, fogs, wells, creatures, etc on the starting space. All features of the game are set up in this operation. Game phase changes to playing.

## 3. In Game

**Operation:** LegendsOfAndor::takeTurn(action: Action[])
**Scope:** Player, Action
**New:** N/A
**Messages:** Player :: {promptOutOfHours_e; promptConfirmOvertimeHours;}
**Description:** Updates the state of the board, based on the user's decision (ie Move Player, Pass, End Day, Battle, Move Prince Thorald)
**Pre:** playerTurn == true
**Post:** The system receives action input from the player and if the player state allows for that action to completed successfully, the system will update the game board and player stats. If the player has enough willpower points to complete their turn in overtime hours, the system will output a “Confirm Overtime Hours” to the player. If there are no hours left in the day, the system will output an “Out of hours” message. If player completes turn and no hours are left in day, then system places player’s hero in sunrise box.

**Operation:** LegendsOfAndor:: openFeatures(Chat, Card: Legend/Event#, Hero: h)
**Scope:** Chat History, Card, Player
**New:** String: possible new chats
**Messages:** Player :: {promptChatWindow, showCard, showHeroBoard}

**Post:** The openFeatures operation allows player to interact with the game instance to view chatWindow to check for chat updates, view current legend, and check other heroes' stats (item owned, strength points, willpower points). All these features are folded tabs on the main game board.

**Operation:** LegendsOfAndor:: completeDay()
**Scope:** Game, Narrator, Player, Map, Space,
**New:** newSpace: creatureSpace
**Messages:** Player:: {promptNewDay}
**Pre:** last player to end their day in a turn triggers this operation
**Post:** This is an extension operation of the end day operation. The complete day is only triggered when the last player to finish their day chooses to end his day. This operation also informs the system to refresh all wells with no one standing on the well, move creatures, advance the narrator. Note that only one creature is allowed on one space, if two creatures are moving onto the same space, one of the creatures would advance to the next adjacent space following the order(gors, skrals, wardraks, trolls)

## 3.1 Menu

**Operation:** LegendsOfAndor:: openMenu()
**Scope:** Player, Game
**New:** N/A
**Messages:** Player :: {promptSaveGame; promptQuitGame; promptEndGame;}
**Post:** This operation allows all players to have access to all the functions we have on the game menu, including endGame, saveGame.

**Operation:** LegendsOfAndor:: quitGame()
**Scope:** Game, Players
**Messages:** OtherPlayer :: {promptloggout; promptGameLobby;}
**Post:** The quitGame operation allows a player to quit an existing game. All other players in the game will be notified with a player left message if this operation is triggered. The remaining players may wait for another player join and continue with the game. The game state changes to can join, and system updates the AvailableJoinGame[].

**Operation:** LegendsOfAndor:: endGame()
**Scope:** Game, Players
**Messages:** OtherPlayer :: {promptloggout; promptGameLobby;}
**Post:** The endGame operation allows a player to end an existing game. A vote has to be passed for this operation to be triggered. This operation deletes the game instance within the database.

**Operation:** LegendsOfAndor::saveGame()
**Scope:** Game, Player, Database
**Messages:** Player :: {promptSaveGame}
**Post:** The saveGame operation allows the player to save the current game state. System stores this instance in database with the time. When the game is rejoined by all players, the most recent saved instance of the game is opened.

## 4. Move (Fight separated)

**Operation:** LegendsOfAndor::endTurn()
**Scope:** Game, Player
**New:** N/A
**Messages:** Player :: {promptNextTurn;}
**Pre:** player's turn == true
**Post:** The System ends the turn of the current Player and moves on to the next Player in order.

**Operation:** LegendsOfAndor::endDay()
**Scope:** Game, Player
**New:** N/A
**Messages:** Player :: {promptSunriseBox;}
**Pre:** It must currently be the Player's turn.
**Post:** The System adds the Player to the list of Players in the sunrise box and changes the hero's *inSunriseBox* boolean to True. This means the player cannot do any game actions such as move their hero or free actions such as use items until they leave the sunrise box. The Player is added to end of list of next day's turn order in the sunrise box.

**Operation:** LegendsOfAndor::movePrince(s: space)

**Scope:** Game,Player, Token
**New:** N/A
**Messages:** Player :: {promptMove;}
**Pre:** Player must be on the same space with Prince Thorald
**Post:** The movePrince operation is only available when the player is on the same space as the prince. The player may carry the prince and use the Move operation to move with Prince Thorald to complete the task.

**Operation:** LegendsOfAndor:: Move(s: Space)
**Scope:** Game, Player, Map, Space
**New:** N/A
**Messages:** Player: {promptExceedDayHours_e;}
**Pre:** Must be the player's turn and the space must be a valid space
**Post:** The space that the player would like to move to is passed to system via the space parameter. The system will compute how many hours it will take the player to move from their current space to the new space and increase their hour tracker by this amount. If there are not enough hours, the system will output a "Not enough hours" message.

**Operation:** LegendsOfAndor::pass()
**Scope:** Player, Game
**New:** N/A
**Messages:** Player :: {promptNextTurn;}
**Pre:** It must currently be the Players turn if they choose to pass
**Post:** Players turn is passed and their hour tracker is increased by 1 by the System. The turn is now moved on to the next player in the order.

# 5. Free Actions (Trade separated)

**Operation:** LegendsOfAndor::activateFog()
**Scope:** Player, Game, Token
**New:** N/A
**Messages:** Player::{clearFog; shieldIneffective_e; promptUseItem;}
**Pre:** Player located on tile with a fog token

**Post:** Once the fog token is activated, the system will modify the game state based on what is unveiled and will remove the fog token unless the player is able to combat activation with a special article such as a shield.

**Operation:** LegendsOfAndor::emptyWell()
**Scope:** Player, Game, Token
**New:** N/A
**Messages:** Player ::{wellEmptied; passingThrough_e;}
**Pre:** Player must be on the same space as an empty well
**Post:** The Player empties the well and receives 3 willpower points. The well will now remain empty until sunrise when it's refreshed by the System unless a player is on the space with the well on it at this time.

**Operation:** LegendsOfAndor::pickUpGD(gold: int, gemstone: int)
**Scope:** Player, Game, Article
**New:** N/A
**Messages:** Player :: {goldgemAdded;}
**Pre:** Player must be stopped on a space with gold or gemstones in order to pick it up
**Post:** System updates the amount of gold or gemstones on the player's board and removes this gold/gemstones from the game board.

**Operation:** LegendsOfAndor::buyItem(Article : item)
**Scope:** Player, Game, Article
**New:** N/A
**Messages:** Player :: {itemAdded; insufficientFunds_e;}
**Pre:** Player must have enough gold to buy the item they select and they must be on the same space (as either the witch, merchant or dwarf mine) to make a purchase.
**Post:** The item bought is distributed to the players (purchasers) game board.

**Operation:** LegendsOfAndor::useItem(Article : item)
**Scope:** Player, Game
**New:** N/A
**Messages:** Player :: {cannotUseItem_e;}

**Pre:** Player must possess article on their hero board.
**Post:** The Player uses an item on their gameboard and the System changes the game state to reflect this use.

## 6. Trade

**Operation:** LegendsOfAndor::displayOwn(itemList: mylist, playList: playerlist)
**Scope:** Players, Game, Map
**New:** newTradeItem : Article
**Messages:** Players {promptTrade; promptItemOwn;}
**Pre:** Players must be on same space or either must possess a falcon
**Post:** This operation allows players to check their items owned and with the option to trade with a selected player if they satisfy the precondition.

**Operation:** LegendsOfAndor::selectTradeItem(Article : item)
**Scope:** Player, Article
**New:** N/A
**Messages:** {itemTradeInvalid_e; sendItem;}
**Pre:** item.own == true
**Post:** The Player chooses which article they wish to trade and informs the System of their choice.

**Operation:** LegendsOfAndor::replyTrade(decision);
**Scope:** Players, Article
**New:** N/A
**Messages:** {promptTradeResponse;}
**Pre:** A trade must have been proposed by another player
**Post:** The player (tradee) informs the System if they accept or reject the trade proposed by the other Player (trader).

## 7. Fight

**Operation:** LegendsOfAndor::joinFight(c: creature)
**Scope:** Players, Game, Creature

**New:** newFight: Fight
**Messages:** Player: {promptRollDice; msgWinLose; promptInviteFight;}
**Pre:** Players and Creature must be on the same space or the Player must be an archer on an adjacent space or possess a bow on an adjacent space
**Post:** This operation allows any player to join a fight when a fight is available. This will create a new Fight instance and create an association between the Player(s) and the Creature they are battling. The Player will then be prompted to roll their dice and/or invite another hero. The system will compute the battle values of the Player(s) and Creature involved and will output the result in a Win/Lose message. System updates the player's time hour tracker.

**Operation:** LegendsOfAndor::leaveFight()
**Scope:** Fight, Player
**New:** N/A
**Messages:** Player :; {msgHeroLeft;}
**Post:** This operation allows any player in a fight to leave a fight when they want. An output message would be sent to all other players in the current fight.

**Operation:** LegendsOfAndor::inviteHero(h: Hero)
**Scope:** Fight, Player
**New:** N/A
**Messages:** Player :: {promptInviteResponse;}
**Pre:** Hero must be on the same space as the Player and the Creature OR be an archer on an adjacent space OR possess a bow on an adjacent space
**Post:** An association is created between the fight and the invited Hero. The system increases the hour tracker by 1 for every battle round. This operation allows a player to invite players to join a fight under the precondition. The output can either be accepted or declined.

**Operation:** LegendsOfAndor::rollDice()
**Scope:** Player, Game
**New:** N/A
**Messages:** Player :: {promptDiceValue; noMoreRolls_e; prompUseItem;}
**Pre:** It must currently be the players turn who is rolling the dice.
DiceItem.available == true

**Post:** The final roll of the dice or decided stopping value is used by the Player in their turn (number of rolls each player has/when they can stop rolling varies depending on hero). In an exceptional case, a hero may use item after rolling dice. The output of this operation would be the product of promptUseItem and DiceValue.

**Operation:** LegendsOfAndor::useItem(Article[]: itemList)
**Scope:** Fight, Player, Article
**New:** N/A
**Messages:** Player :; {chooseItem;}
**Pre:** Player must be in possession of item and item must be usable in a fight
**Post:** If the player would like to use an item in a fight, the system updates the Player's battle value depending on which item it is.

**Operation:** LegendsOfAndor::distributeRewards(Gold: int, Willpower: int)
**Scope:** Game, Players
**New:** N/A
**Messages:** Player :: {distributionResponse;}
**Pre:** The players unanimously agree on how they want the rewards in the system to be distributed
**Post:** The System distributes the gold or willpower points as specified by the players, unanimously.

# 8. Event Card

**Operation:** LegendsOfAndor:: replyEventCard(EventCard #n, EventVote())
**Scope:** Card, Player
**New:** newCard: EventCard #n
**Messages:** Player :: {promptTask; promptEventCardResponse; }
**Post:** The replyEventCard operation informs all players that there is an event happening. The system would prompt a *collaborative decision* request for the event and output the response to the current event card.

# 9. Communication

**Operation:** LegendsOfAndor::createChatMessage(text: String)

**Scope:** Player, ChatHistory, ChatMessage
**New:** chatMessage: ChatMessage
**Messages:** Player::{displayChat;}
**Post:** A ChatMessage instance chatMessage is created with the given text and the chatMessage.

## 10. Collaborative Actions

**Operation:** LegendsOfAndor::initiateVotingSession()
**Scope:** Decision, Player
**New:** newVote: vote String: voteName
**Messages:** Player :: {promptSendVote;}
**Post:** The initiateVoteingSession operation allows every player on the game to vote and make collaborative decisions. A vote session would be created and sent to all other players when this operation is triggered.

**Operation:** LegendsOfAndor::sendVoteDesion(self : Player, option : int)
**Scope:** Decision, Player
**New:** N/A
**Messages:** Player :: {promptVotingResult;}
**Post:** The vote decision operation takes all votes from all players and outputs the final decision as either a pass or a fail. The only exception is when there is a tie, the game continues.

**Operation:** LegendsOfAndor::makeCollaborativeDecision(request : Request, p : Player[])
**Scope:** Decision, Game, Player
**New:** newRequest: request
**Messages:** Player :: {promptDecisionToMake; promptDecision;}
**Pre:** CollaborativeDec.exist() == true
**Post:** This operation will be available when the there is a collaborative decision that has to be made according to the *event cards*. Players have to make the decision collaboratively with this operation.

**Operation:** LegendsOfAndor::responseCollaborativeDecision(decision)
**Scope:** Decision, Game, Player

**New:** N/A
**Messages:** Player :: {promptFinalDecision;}
**Post:** The system would process the final decision made from the collaborative decision and execute the desired decision.

## 11. Game Conclusion Process

**Operation:** LegendsOfAndor::GameConclusion(Game)
**Scope:** Game
**New:** N/A
**Messages:** Player :: {promptGameWinOrLose;}
**Post:** The effect of this operation is to check all winning and losing conditions of each game instance and output the final result of the game if the game is a win or a loss.