|  |  |
| --- | --- |
| Flow of Events for the Overall System use-case | |
| Objective | To register players and run the game until there is a winner |
| Precondition | The program must be run |
| Main Flows | 1. The player registration starts as system is launched 2. The players take turns 3. The players perform actions on squares 4. The players are eliminated when balance reaches zero 5. The players are eliminated till there is only one 6. The program ends |
| Alternative Flows | At 2, the players may select to quit. In this case, the game will be closer to ending. |
| Post- Condition | A winner must be displayed |

|  |  |
| --- | --- |
| Flow of Events for the Register Player use-case | |
| Objective | To register a new player in the game data |
| Precondition | The number of players already registered must not exceed 8 |
| Main Flows | 1. The player registration starts as system is launched 2. The player is asked to provide name 3. The player enters a name 4. The player is initialised as a Player object 5. The player list is updated 6. The next player starts registration |
| Alternative Flows | At 3, the player may enter a name that is invalid. In this case the player will be asked to enter re-enter a valid name.  At 3, the player enters “start” as a name the registration will cancel and gameplay will start.  At 5, there may already be the maximum number of players already registered in the game. In this case the registration is stopped.  At 6, the player list reaches 8, the maximum amount, in this case the gameplay will start. |
| Post- Condition | The player is registered to the game and the number of players is updated |

|  |  |
| --- | --- |
| Flow of Events for the Take Turn use-case | |
| Objective | For the player to be able to take their turn when it is their go in the game |
| Precondition | It must be the player’s turn within the game’s turn counter |
| Main Flows | 1. The game displays actions that the player can take. 2. The player selects the action/actions they wish to take during their turn 3. The player ‘rolls the dice’ and ends their turn |
| Alternative Flows | At 2, the player enters and invalid choice. In this case they will have to re-enter a new choice until it is valid |
| Post- Condition | The player has taken their turn and the player turn counter is updated |

|  |  |
| --- | --- |
| Flow of Events for the Roll Dice use-case | |
| Objective | For the player to roll the dice to move to their next position |
| Precondition | It must be the players turn for them to be able to roll the dice |
| Main Flows | 1. The player selects to roll the dice 2. The dice produces a number to show the player how many locations on the board they will move 3. The player’s position is updated 4. The board game displays the location 5. The game displays actions to be taken. 6. The player selects the action to be taken |
| Alternative Flows | At 4, the player has landed at a square with no actions. In this case main flow 6 is skipped |
| Post- Condition | The player turn counter is updated and the next player’s turn begins |

|  |  |
| --- | --- |
| Flow of Events for the Acquire Priority use-case | |
| Objective | To acquire an already existing priority within the boardgame |
| Precondition | This priority must be unowned |
| Main Flows | 1. The player lands on priority 2. The player wants to acquire priority 3. The player purchases the priority 4. The player’s balance is updated 5. The player priority list is updated |
| Alternative Flows | At 2, the player may not have enough funds. In this case this player is unable to acquire the property.  At 5, the player has acquired a full action area. In this case, the player will be notified and allowed to tackle it. |
| Post- Condition | The priority is acquired and players that now land on it owe a contribution |

|  |  |
| --- | --- |
| Flow of Events for the Contribution Transaction use-case | |
| Objective | The player must pay the contribution |
| Precondition | The player must have landed on an already owned priority by another player- property owned player |
| Main Flows | 1. The player lands on a priority 2. The player pays the contribution 3. The player’s contribution is transferred to the Priority Owner. 4. The balance of the player and the priority owner are updated |
| Alternative Flows | At 2, the player may not have sufficient funds. The player will be removed from the game |
| Post- Condition | The contribution is paid |

|  |  |
| --- | --- |
| Flow of Events for the Tackle Priority Transaction use-case | |
| Objective | To develop priority |
| Precondition | The player must own the entire action area which the priority they want to update is included in |
| Main Flows | 1. The player chooses to tackle priority 2. The player selects priority 3. The player selects how many steps 4. The priority is updated |
| Alternative Flows | At 1, there is no priority developments available. In this case, the game will print no priorities available.  At 2, the player may not fully own the action area. In this case the transaction is cancelled  At 3, the player may have insufficient funds to tackle the priority. In this case the player is unable to tackle the priority and main flow 4 is skipped. |
| Post- Condition | The priority is updated |

|  |  |
| --- | --- |
| Flow of Events for the Remove Player use-case | |
| Objective | To remove player from the board game |
| Precondition | It must be the players turn |
| Main Flows | 1. The player has selected to remove player 2. The removed player’s priorities are reset and returned 3. The player list is updated |
| Alternative Flows | At 2, the player may have no priorities owned. In this case there may be no priorities to be reset.  At 3, the player list may now only contain one player. In this case the game will end. |
| Post- Condition | The player is removed, and the total player amount is decreased. |

|  |  |
| --- | --- |
| Flow of Events for the Pass Go Transaction use-case | |
| Objective | To pass starting position of the board |
| Precondition | It must be the players turn |
| Main Flows | 1. The player lands on or passes go (start square) 2. The player is notified they passed the starting square 3. The set amount is added to the player’s balance 4. The player’s new balance is printed |
| Alternative Flows |  |
| Post- Condition | The player passes go |