Name___Robbie Reinhardt____________ _____________________/50

[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

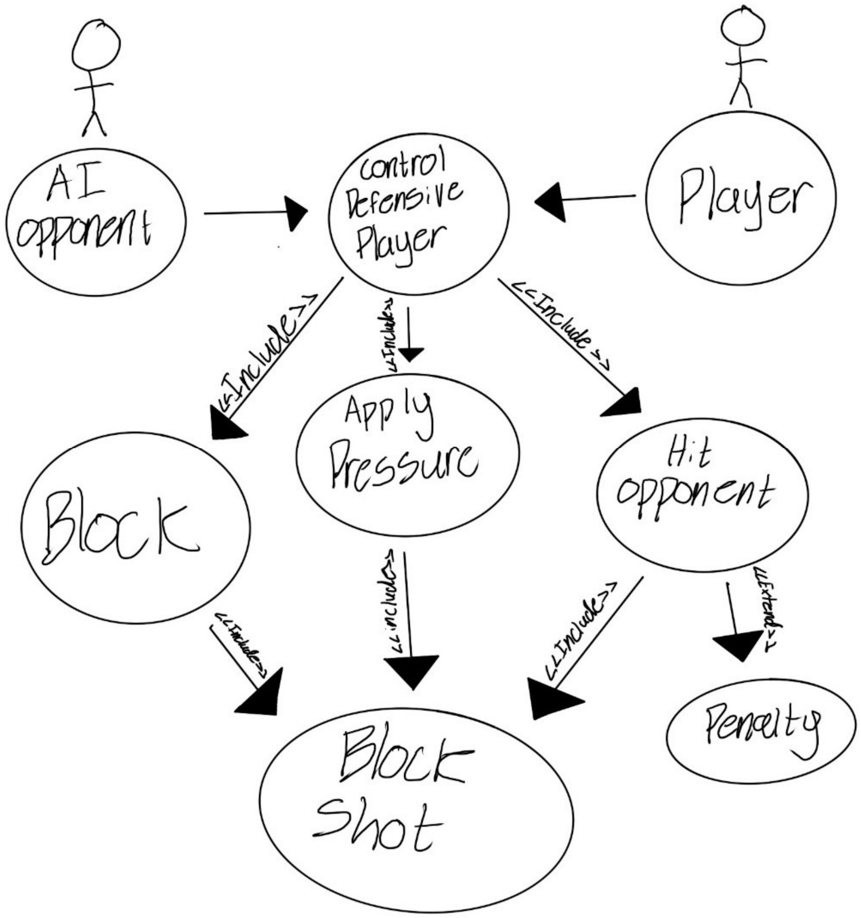
1. Brief introduction __/3

My feature for the Retro Hockey game is the Automated Defense System (ADS). This system is responsible for controlling the defensive players on the AI-controlled team. When the opposing team has the puck, the ADS takes over the non-puck-carrying defensive players to position them strategically and react to offensive plays. This system must provide a realistic and challenging AI opponent by ensuring that the defensive players intelligently block shots, intercept passes, and pressure the puck carrier. The ADS needs to balance aggressive play with maintaining proper defensive positioning to avoid leaving open lanes to the goal.

My responsibilities include creating logic that determines which defensive player to control, how to position them relative to the puck and the net, and what actions to take in response to the offensive team's movements. This system is crucial for the "Road to the Stanley Cup" mode, as it will be scaled in difficulty to provide a progressively harder challenge for the player.

2. Use case diagram with scenario __14

Use Case Diagrams



Scenarios

Scenario 1 (1st Use Case Diagram):

Name: Control Defensive Player

Summary: The automated defense system takes control of a defensive player to respond to the puck carrier's movements.

Actors: Automated Defense System

Preconditions: The offensive team has possession of the puck. The game is in a playable state.
Basic Sequence: The game detects that the puck is in the offensive zone.
The automated defense system identifies the puck carrier and the nearest defensive player. The system calculates a defensive strategy based on the puck carrier's position and movement. The system commands the defensive player to move into a blocking or pressuring position.

Exceptions:
-Step 1: The puck carrier enters the defensive zone from an unexpected angle (e.g., from behind the net).
-Step 2: The system's logic recalculates the defensive player's movement to cut off the puck carrier's new path to the goal, overriding the initial command.

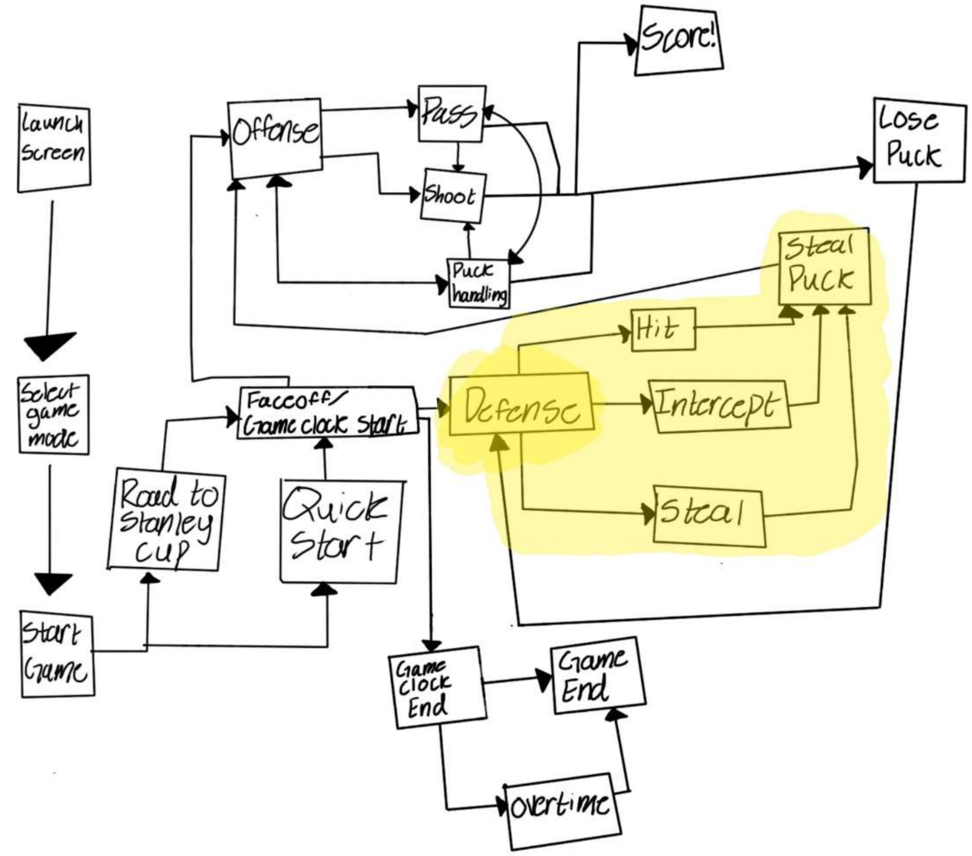
Post conditions: The defensive player is positioned to effectively challenge the puck carrier and defend the goal.

Priority: 1
* *The priorities are 1 = must have, 2 = essential, 3 = nice to have.
ID: AD1

3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

[Get the Level 0 from your team. Highlight the path to your feature]

Data Flow Diagrams



Process Descriptions

Automated Defense System

This process is designed to manage all defensive players for the AI-controlled team. Its primary function is to analyze the game state and command the most relevant defensive player to react appropriately.

WHILE the puck is not in your team's possession:

Identify Puck Position: The system first determines the exact location of the puck on the ice. Identify All Player Positions: It then collects the coordinates of all players from both the opponent's and your team's side.

Find Nearest Defensive Player: Based on the puck's position, the system identifies the defensive player who is in the best position to challenge the puck carrier.

Determine Defensive Strategy: The AI's decision-making logic is applied here, adapting to the game's difficulty level:

IF the difficulty is "Easy": The AI calculates a simple path for the defensive player to move directly between the puck and the goal.

ELSE IF the difficulty is "Medium": The AI's strategy becomes more nuanced, calculating a position relative to the puck and the nearest opponent.

ELSE IF the difficulty is "Hard": The AI uses an advanced algorithm to anticipate the opponent's movement and potential passing lanes, positioning the player to cut off plays before they develop.

Command Defensive Player: The system sends a command to the selected defensive player, instructing them to move to the calculated position to apply pressure and attempt to steal the puck.

END WHILE: This loop concludes once the puck is successfully stolen by the defensive team, at which point the game state transitions to offense.

4. Acceptance Tests _______9

The acceptance tests for this feature will focus on validating the ADS's strategic positioning and responsiveness. These tests will be automated using Unity’s automated testing system to ensure consistency and a predictable, yet challenging, AI. We will focus on unit tests for specific functions and boundary conditions

Defensive Positioning Test:

- **Test Name:** TestDefensivePlayerPositioning
- **Description:** This test will verify that defensive players move to the correct strategic position relative to the puck.
- **Input:** A static puck carrier is placed at a specific X,Y coordinate (e.g., (10, 5)).
- **Expected Output:** The nearest defensive player should move to a predictable "covering" position (e.g., between the puck carrier and the net), based on a predefined formula. The defensive player's final position should be within a small tolerance of the target coordinates.
- **Boundary Conditions:**
  - Test puck carrier positioning at the edge of the rink or directly in front of the net to ensure the ADS logic handles these extreme cases correctly.
  - Test multiple puck carriers to ensure the system selects and positions the correct defensive player to counter the most dangerous threat.

Test Name	Preconditions	Input	Expected Output	Notes
Test 1: Defensive Positioning	Puck is controlled by the offensive player at (10, 5). Defensive player is at (10, -5).	Start_Position_X = 10, Start_Position_Y = 5.	The defensive player moves to (10, 0) with a margin of error of +/- 0.5.	The system must calculate a path and position the defensive player between the puck carrier and the goal, even if the puck carrier is not moving.
Test 2: Defensive Positioning (Boundary Case)	Puck is controlled by the offensive player at the corner of the rink, (95, 45). Defensive player is at (95, -45).	Start_Position_X = 95, Start_Position_Y = 45.	The defensive player moves to a position to cut off the shortest path to the net, such as (95, 30) with a margin of error of +/- 0.5.	The system must handle non-standard positions and not glitch or get stuck on the rink boundaries.
Test 3: Pass Interception	Offensive player 1 at (15, 0). Offensive player 2 is at (0, 0). A pass is initiated.	Pass_Start_X = 15, Pass_Start_Y = 0. Pass_End_X = 0, Pass_End_Y = 0. Defensive player at (7.5, 0).	Puck_Status changes from "in motion" to "intercepted". The defensive player's Has_Puck flag becomes True.	The defensive player must intercept the puck if they are within a specific range of the pass's path.
Test 4: Pass Interception (Bad Input)	Offensive player 1 at (15, 0). Offensive player 2 is at (0, 0). A pass is initiated.	Pass_Start_X = 15, Pass_Start_Y = 0. Pass_End_X = 0, Pass_End_Y = 0. Defensive player is too far away to intercept, at (15, 5).	No Puck_Status change occurs. The Has_Puck flag for the defensive player remains False. The defensive player's movement is not altered by the pass.	The system should not attempt an interception if the player is not in a plausible interception zone.

Pass Interception Test:

- **Test Name:** TestPassInterception
- **Description:** This test verifies that a defensive player attempts to intercept a pass within their effective range.
- **Input:** A pass is initiated between two offensive players. A defensive player is positioned within the pass's path.
- **Expected Output:** The defensive player's collider should intersect the puck's path. An "intercept" event should be triggered, and the puck's velocity should be altered, indicating a successful interception. The system should correctly change possession.
- **Boundary Conditions:**
    - Test a pass that is just out of the defensive player's range. The defensive player should not trigger an intercept, and the puck should continue on its path.
    - Test a fast pass that the defensive player cannot physically reach. The system should correctly fail to intercept, but the defensive player should still attempt to get into position.

Test Name	Preconditions	Input	Expected Output	Notes
Test 1: Movement & Speed	Player character is at (0, 0) on an empty rink.	Continuous "move right" input for 3 seconds.	The player's final X_position is > 0. Their final Y_position is 0. The final position must be consistent across multiple test runs.	Verify that the movement speed is consistent and that the player only moves in the intended direction. This is the "regular condition" test.
Test 2: Movement (Bad Input)	Player character at (10, 10).	Simultaneous "move left" and "move right" inputs for 3 seconds.	The player's final position remains at (10, 10).	The system must gracefully handle conflicting inputs without any movement or unexpected behavior.
Test 3: Boundary Collision (In Range)	Player character is at the edge of the rink, at (100, 50).	Continuous "move right" input for 5 seconds.	The player's final X_position remains at 100. Their Y_position remains 50.	The system must not allow the player to move beyond the defined boundary of the rink.
Test 4: Boundary Collision (Out of Range)	Player character is placed at (105, 50) via debug tools.	A "snap to grid" or "collision check" function is executed.	The player's position is immediately adjusted back to (100, 50). The Is_Valid_Position flag becomes True.	The system must have a graceful method for handling an out-of-bounds position. This is similar to the calculator's "MAX_VAL" for divide by zero.

5. Timeline ________/10

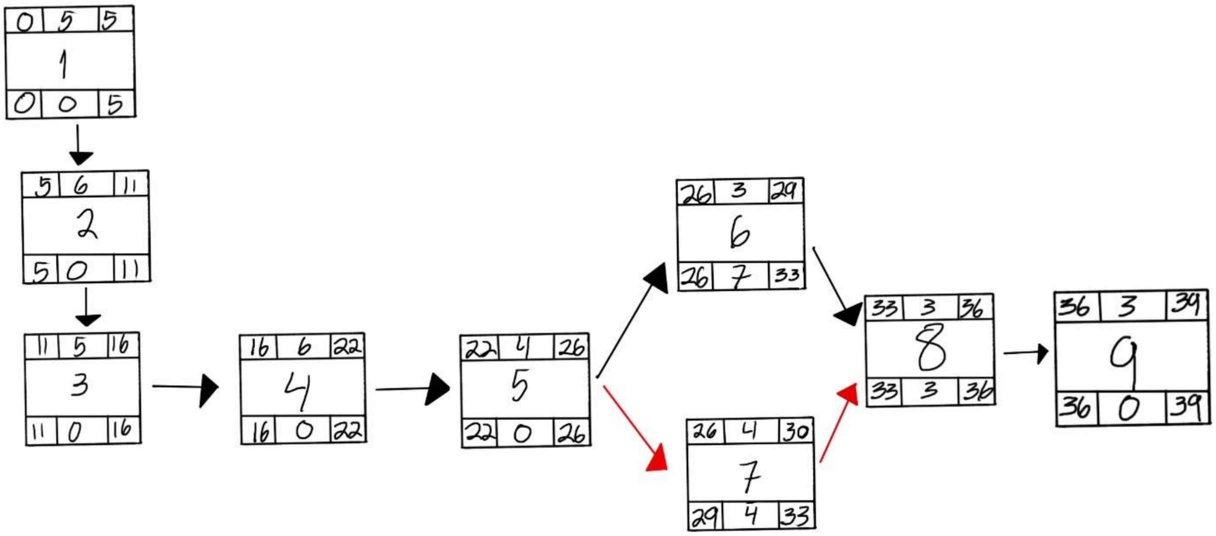
[Figure out the tasks required to complete your feature]

Example:

Work items – Time changed to hours

Task	Duration (hrs)	Predecessor Task(s)
1. Defensive Player selection Logic	5	
2. Basic positioning logic	6	1
3. Puck pressure logic	5	2
4. Pass interception logic	6	3
5. difficulty scaling implementation	4	4
6. Documentation	3	5
7. Unit testing	4	5
8. Integration/Bug Fixing	3	6, 7
9. Artwork/Animations	3	

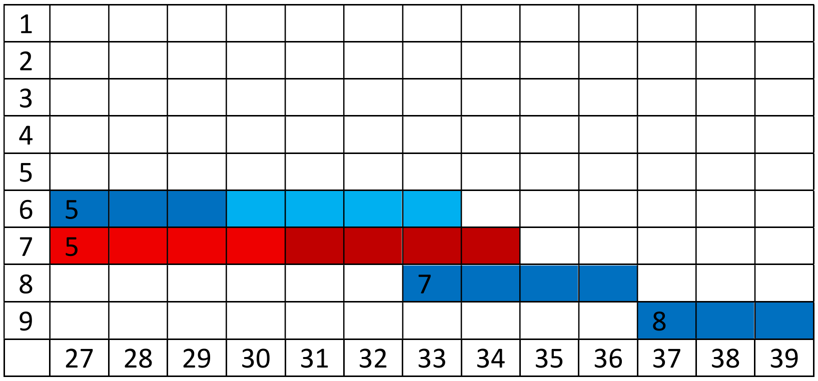
Pert diagram



Gantt timeline

1	■	■	■	■	■																					
2						1	■	■	■	■	■															
3												2	■	■	■	■										
4																	3	■	■	■	■	■				
5																							4	■	■	■
6																										
7																										
8																										
9																										
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Pt.2 of Gantt Chart:



1													
2													
3													
4													
5													
6	🟦 5	🟦	🟦	🔷	🔷	🔷	🔷						
7	🟥 5	🟥	🟥	🟥	🟫	🟫	🟫	🟫					
8							🟦 7	🟦	🟦	🟦			
9											🟦 8	🟦	🟦
	27	28	29	30	31	32	33	34	35	36	37	38	39