Test Case Documentation

Bio-Hazard Cleaning Intelligent Agent System

Date: February 16, 2026
Total Test Cases: 173
Framework: Python unittest + pytest

Section 1: Environment Module Test Cases (43 Tests)

1.1 Initialization & Grid Setup Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
1	ENV-001	Environment creates 20x20 grid	Environment module imported	1. Create Environment(20) 2. Check size attribute 3. Verify grid shape	size=20	grid dimensions 20x20, size attribute = 20	20x20 grid created	☑ PASS
2	ENV-002	Environment creates 50x50 grid	Environment module imported	1. Create Environment(50) 2. Check size attribute 3. Verify grid shape	size=50	grid dimensions 50x50, size attribute = 50	50x50 grid created	☑ PASS
3	ENV-003	Environment creates 100x100 grid	Environment module imported	1. Create Environment(100) 2. Check size attribute 3. Verify grid shape	size=100	grid dimensions 100x100, size attribute = 100	100x100 grid created	☑ PASS
4	ENV-004	Grid initialized with correct dtype	Environment module imported	1. Create Environment(10) 2. Check grid.dtype	size=10	dtype is numpy integer type	dtype verified	☑ PASS

1.2 Inaccessible Area Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
5	ENV-005	All borders marked inaccessible	Environment(20) created	1. Create env 2. Check grid[0,:] 3. Check grid[19,:] 4. Check grid[:,0] 5. Check grid[:,19]	size=20	All border cells have value=2	Borders marked as 2	☑ PASS
6	ENV-006	Academic Building marked inaccessible	Environment(100) created	1. Check grid[1:21, 79:99] region	size=100	All cells in region = 2	Building area = 2	☑ PASS
7	ENV-007	Administrative Building marked inaccessible	Environment(100) created	1. Check grid[69:99, 1:31] region	size=100	All cells in region = 2	Building area = 2	☑ PASS
8	ENV-008	Pond area marked inaccessible	Environment(100) created	1. Check grid[35:61, 35:61] region	size=100	All cells in region = 2	Pond area = 2	☑ PASS
9	ENV-009	Guest House marked inaccessible	Environment(100) created	1. Check grid[40:50, 0:15] region	size=100	All cells in region = 2	Building area = 2	☑ PASS
10	ENV-010	Multiple buildings coexist in 100x100	Environment(100) created	1. Verify all building areas present	size=100	All 5+ building areas marked value=2	Multiple buildings present	☑ PASS

1.3 is_inside_grid() Boundary Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
11	ENV-011	Valid center position returns True	env = Environment(10)	1. Call is_inside_grid((5,5)) 2. Check return value	position=(5,5), size=10	Returns True	True returned	☑ PASS
12	ENV-012	Valid corner positions return True	env = Environment(10)	1. Call is_inside_grid((0,0)) 2. Call is_inside_grid((9,9))	positions (0,0) and (9,9)	Both return True	True returned	☑ PASS
13	ENV-013	Position outside right boundary returns False	env = Environment(10)	1. Call is_inside_grid((5,10)) 2. Check return value	position=(5,10), size=10	Returns False	False returned	☑ PASS
14	ENV-014	Position outside bottom boundary returns False	env = Environment(10)	1. Call is_inside_grid((10,5)) 2. Check return value	position=(10,5), size=10	Returns False	False returned	☑ PASS
15	ENV-015	Negative row index returns False	env = Environment(10)	1. Call is_inside_grid((-1,5)) 2. Check return value	position=(-1,5), size=10	Returns False	False returned	☑ PASS
16	ENV-016	Negative column index returns False	env = Environment(10)	1. Call is_inside_grid((5,-1)) 2. Check return value	position=(5,-1), size=10	Returns False	False returned	☑ PASS
17	ENV-017	Boundary positions validated correctly	env = Environment(20)	1. Test (0,0) 2. Test (19,19) 3. Test (19,0) 4. Test (0,19)	size=20	All boundary positions return True	True for all	☑ PASS

1.4 is_accessible() Validation Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
18	ENV-018	Clean accessible area returns True	env = Environment(10), position not marked inaccessible	1. Call is_accessible((5,5)) 2. Check return value	position=(5,5), clean cell	Returns True	True returned	☑ PASS
19	ENV-019	Building area returns False	env = Environment(100)	1. Call is_accessible((5,85)) 2. Check return value	position in building area	Returns False	False returned	☑ PASS
20	ENV-020	Border position returns False	env = Environment(20)	1. Call is_accessible((0,10)) 2. Check return value	position on border	Returns False	False returned	☑ PASS
21	ENV-021	Outside grid returns False	env = Environment(10)	1. Call is_accessible((15,15)) 2. Check return value	position=(15,15), size=10	Returns False	False returned	☑ PASS

1.5 is_bio_hazard() Detection Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
22	ENV-022	Detect bio hazard at position	env with hazard placed	1. Place hazard at (5,5) 2. Call is_bio_hazard((5,5))	position with hazard	Returns True	True returned	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
23	ENV-023	Clean area returns False for bio_hazard	env = Environment(10)	1. Call is_bio_hazard((5,5)) 2. Check return value	clean position	Returns False	False returned	☑ PASS

1.6 is_clean() Detection Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
24	ENV-024	Clean cell detection	env = Environment(10)	1. Call is_clean((5,5)) 2. Check return value	clean position	Returns True	True returned	☑ PASS
25	ENV-025	Cleaned position after cleaning	env with hazard cleaned	1. Clean hazard 2. Call is_clean((5,5))	cleaned position	Returns True	True returned	☑ PASS

1.7 Bio-Hazard Placement Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
26	ENV-026	Place specified number of hazards	env = Environment(20)	1. Call place_bio_hazards(5) 2. Verify count	bio_hazard_count=5	Returns 5, 5 hazards placed	5 hazards placed	☑ PASS
27	ENV-027	Handle zero hazards safely	env = Environment(20)	1. Call place_bio_hazards(0) 2. Check return value	bio_hazard_count=0	Returns 0 safely	0 returned	☑ PASS
28	ENV-028	No duplicate hazard placement	env = Environment(20)	1. Place 5 hazards 2. Verify all unique positions	bio_hazard_count=5	All hazards at unique positions	All unique	☑ PASS
29	ENV-029	Placement only in accessible areas	env = Environment(100)	1. Place hazards 2. Verify in accessible cells	bio_hazard_count=10	All hazards in accessible areas (value=0)	All accessible	☑ PASS
30	ENV-030	Place maximum possible hazards	env = Environment(20)	1. Calculate max accessible 2. Place max 3. Verify return count	full grid	Correct count returned	Count verified	☑ PASS
31	ENV-031	Return zero when space unavailable	env full of hazards	1. Call place_bio_hazards(10) 2. Check return value	bio_hazard_count=10, no space	Returns 0 or available count	0 returned	☑ PASS

1.8 Count Functions Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
32	ENV-032	Count bio hazards zero initially	env = Environment(20)	1. Call count_bio_hazards() 2. Check return value	no hazards placed	Returns 0	0 returned	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
33	ENV-033	Count matches placed hazards	env with 5 hazards placed	1. Call count_bio_hazards() 2. Compare to placed count	bio_hazard_count=5	Returns 5	5 returned	☑ PASS
34	ENV-034	Count clean areas initially	env = Environment(20)	1. Call count_clean_areas() 2. Verify against grid	size=20	Returns correct clean cell count	Count correct	☑ PASS
35	ENV-035	Count decreases after hazard placement	env with hazards placed	1. Get initial count 2. Place hazard 3. Get new count	size=20	New count less than initial	Count decreased	☑ PASS
36	ENV-036	Count inaccessible areas consistent	env = Environment(20)	1. Call count_inaccessible_areas() 2. Verify consistency	size=20	Returns consistent count	Count consistent	☑ PASS
37	ENV-037	Count accessible areas correctly	env = Environment(20)	1. Call count_accessible_areas() 2. Verify calculation	size=20	Returns accessible cell count	Count correct	☑ PASS
38	ENV-038	Sum of areas equals grid size	env = Environment(20)	1. Count accessible 2. Count inaccessible 3. Sum them	size=20	accessible + inaccessible = 400	Sum verified	☑ PASS
39	ENV-039	Get clean area coordinates	env = Environment(20)	1. Call get_clean_area_coordinates() 2. Verify list	no hazards	Returns list of clean positions	List returned	☑ PASS

1.9 Grid Getter Methods Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
40	ENV-040	get_grid() returns copy	env = Environment(10)	1. Get grid copy 2. Modify copy 3. Check original unchanged	size=10	Original grid unaffected	Copy independent	☑ PASS
41	ENV-041	get_grid() has correct dimensions	env = Environment(20)	1. Get grid copy 2. Check shape	size=20	Shape is (20,20)	Shape verified	☑ PASS
42	ENV-042	Get bio hazard coordinates	env with hazards	1. Call get_bio_hazard_coordinates() 2. Verify list	hazards placed	Returns list of (r,c) tuples	List correct	☑ PASS
43	ENV-043	Get inaccessible coordinates	env = Environment(20)	1. Call get_inaccessible_coordinates() 2. Verify count matches	size=20	Count matches inaccessible areas	Count verified	☑ PASS

Section 2: Agent Module Test Cases (46 Tests)

2.1 Initialization Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
1	AGT-001	Agent starts at given position	Agent module imported	1. Create Agent((10,10)) 2. Check current_position	start_position=(10,10)	current_position=(10,10)	Position set	☑ PASS
2	AGT-002	Agent initializes as active	Agent module imported	1. Create Agent((10,10)) 2. Check active attribute	start_position=(10,10)	active=True	active=True	☑ PASS
3	AGT-003	Stop reason initializes as None	Agent module imported	1. Create Agent((10,10)) 2. Check stop_reason	start_position=(10,10)	stop_reason=None	stop_reason=None	☑ PASS
4	AGT-004	Visited positions initialized	Agent module imported	1. Create Agent((10,10)) 2. Check visited_positions	start_position=(10,10)	Contains start position	Start position visited	☑ PASS

2.2 Position Update Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
5	AGT-005	Update position changes current	agent = Agent((10,10))	1. Call update_position((11,10)) 2. Check current_position	new_position=(11,10)	current_position=(11,10)	Position updated	☑ PASS
6	AGT-006	Steps incremented on move	agent = Agent((10,10))	1. Call update_position((11,10)) 2. Check steps_taken	new_position=(11,10)	steps_taken=1	steps=1	☑ PASS
7	AGT-007	Position added to visited set	agent = Agent((10,10))	1. Call update_position((11,10)) 2. Check visited_positions	new_position=(11,10)	(11,10) in visited_positions	Position in visited	☑ PASS
8	AGT-008	Position added to path	agent = Agent((10,10))	1. Call update_position((11,10)) 2. Check path list	new_position=(11,10)	(11,10) in path	Position in path	☑ PASS
9	AGT-009	Path maintains chronological order	agent = Agent((10,10))	1. Move to (11,10) 2. Move to (12,10) 3. Check order	moves in sequence	Path in correct order	Path ordered	☑ PASS
10	AGT-010	Multiple movements tracked	agent = Agent((10,10))	1. Move 5 times 2. Check all tracked	5 sequential moves	All 5 positions tracked	Tracked correctly	☑ PASS

2.3 Waste Collection Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
11	AGT-011	Waste counter incremented	agent = Agent((10,10))	1. Call collect_waste() 2. Check waste_collected	none	waste_collected=1	Count=1	☑ PASS
12	AGT-012	Multiple collections tracked	agent = Agent((10,10))	1. Call collect_waste() 3 times 2. Check counter	3 collections	waste_collected=3	Count=3	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
13	AGT-013	collect_waste() returns None	agent = Agent((10,10))	1. Call collect_waste() 2. Check return value	none	Returns None	None returned	☑ PASS
14	AGT-014	Waste count in statistics	agent after collecting	1. Collect waste 2. Get statistics 3. Check waste_collected	2 collections	stats['waste_collected']=2	Count in stats	☑ PASS

2.4 Termination/Stop Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
15	AGT-015	Stop sets active to False	agent = Agent((10,10))	1. Call stop("Hit border") 2. Check active	reason="Hit border"	active=False	active=False	☑ PASS
16	AGT-016	Stop sets stop reason	agent = Agent((10,10))	1. Call stop("Hit border") 2. Check stop_reason	reason="Hit border"	stop_reason="Hit border"	Reason set	☑ PASS
17	AGT-017	Different stop reasons	agent = Agent((10,10))	1. Test various reasons 2. Verify each	multiple reasons	All reasons set correctly	Reasons correct	☑ PASS
18	AGT-018	Stop reason preserved once set	agent after stop called twice	1. Call stop("first") 2. Call stop("second") 3. Check reason	two stop calls	stop_reason="first"	First preserved	☑ PASS

2.5 Visited Positions Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
19	AGT-019	Starting position marked visited	agent = Agent((10,10))	1. Check has_visited((10,10))	start_position=(10,10)	Returns True	True	☑ PASS
20	AGT-020	Position marked after movement	agent after move	1. Move to (11,10) 2. Check has_visited((11,10))	new_position=(11,10)	Returns True	True	☑ PASS
21	AGT-021	Multiple positions tracked	agent after 3 moves	1. Move to 3 positions 2. Check each visited	3 positions	All return True	All visited	☑ PASS
22	AGT-022	Unvisited position returns False	agent = Agent((10,10))	1. Check has_visited((15,15))	unvisited position	Returns False	False	☑ PASS

2.6 Path Methods Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
23	AGT-023	Path length correct	agent after 3 moves	1. Get path 2. Check length	3 moves	len(path)=4	Length=4	☑ PASS
24	AGT-024	Path maintains chronological order	agent after 3 moves	1. Get path 2. Verify order	sequential moves	Path in move order	Order correct	☑ PASS
25	AGT-025	get_path() returns copy	agent = Agent((10,10))	1. Get path copy 2. Modify 3. Get path again	none	Original path unchanged	Copy independent	☑ PASS
26	AGT-026	Steps equals path length minus 1	agent after 5 moves	1. Check steps_taken 2. Check path length	5 moves	steps=4, len(path)=5	Relationship correct	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
27	AGT-027	Visited positions are unique path	agent after moves	1. Compare visited_positions with unique(path)	multiple moves	visited == set(path)	Sets match	☑ PASS

2.7 Statistics Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
28	AGT-028	Statistics returns dictionary	agent = Agent((10,10))	1. Call get_statistics() 2. Check type	none	Type is dict	Dict returned	☑ PASS
29	AGT-029	Statistics has required keys	agent = Agent((10,10))	1. Get statistics 2. Check keys	none	Has steps_taken, waste_collected, stop_reason	Keys present	☑ PASS
30	AGT-030	Initial statistics values	agent = Agent((10,10))	1. Get initial statistics	fresh agent	steps=0, waste=0, reason=None	Values correct	☑ PASS
31	AGT-031	Statistics updated after movement	agent after 2 moves	1. Get statistics	2 moves	stats['steps_taken']=2	Updated	☑ PASS
32	AGT-032	Statistics updated after collection	agent after collecting	1. Get statistics	collections made	stats['waste_collected'] updated	Updated	☑ PASS
33	AGT-033	Statistics updated after stop	agent after stop	1. Get statistics	stop called	stats['stop_reason'] set	Updated	☑ PASS
34	AGT-034	Statistics all accurate combined	agent after all operations	1. Move, collect, stop 2. Get stats	complex operations	All stats correct	All correct	☑ PASS
35	AGT-035	get_current_position works	agent = Agent((10,10))	1. Call get_current_position()	start position	Returns (10,10)	Position returned	☑ PASS
36	AGT-036	Path copy is independent	agent = Agent((10,10))	1. Get path 2. Modify 3. Check original	copy operations	Original unchanged	Independent	☑ PASS

2.8 Edge Cases Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
37	AGT-037	Large number of movements	agent = Agent((10,10))	1. Execute 100 moves 2. Check tracking	100 moves	All tracked correctly	Tracked	☑ PASS
38	AGT-038	Large waste collections	agent = Agent((10,10))	1. Collect waste 50 times 2. Check count	50 collections	waste_collected=50	Count=50	☑ PASS
39	AGT-039	Position tuple and list	agent = Agent((10,10))	1. Update with tuple 2. Update with list	both formats	Both work equally	Both work	☑ PASS
40	AGT-040	State consistency	agent through operations	1. Move, collect, stop 2. Check consistency	all operations	State valid throughout	Consistent	☑ PASS
41	AGT-041	Path includes starting position	agent = Agent((10,10))	1. Check path 2. Verify start included	fresh agent	path[0] = start_position	Start in path	☑ PASS
42	AGT-042	Zero visited positions	agent = Agent((0,0))	1. Call has_visited((0,0))	position (0,0)	Returns True	True	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
43	AGT-043	Negative coordinates valid	agent = Agent((-5,-5))	1. Create with negative 2. Check position	negative coords	Accepted and tracked	Accepted	☑ PASS
44	AGT-044	Same position multiple times	agent after revisits attempted	1. Visit same position twice 2. Check visited	same position	Only counted once in visited	Counted once	☑ PASS
45	AGT-045	Statistics consistency	agent through operations	1. Perform actions 2. Compare stats	operations	stats consistent with actual	Consistent	☑ PASS
46	AGT-046	Rapid state changes	agent with quick operations	1. Move multiple times rapidly 2. Check state	rapid moves	State valid and complete	Valid	☑ PASS

Section 3: Movement Module Test Cases (34 Tests)

3.1 Initialization Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
1	MOV-001	Movement has environment	movement = Movement(env, agent)	1. Check movement.environment	environment object	environment reference set	Set correctly	☑ PASS
2	MOV-002	Movement has agent	movement = Movement(env, agent)	1. Check movement.agent	agent object	agent reference set	Set correctly	☑ PASS

3.2 Valid Move Direction Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
3	MOV-003	Right direction move valid	movement set up, position (10,10)	1. Call is_move_valid((10,10), (10,11), set())	from (10,10) to (10,11)	Returns True	True	☑ PASS
4	MOV-004	Left direction move valid	movement set up, position (10,10)	1. Call is_move_valid((10,10), (10,9), set())	from (10,10) to (10,9)	Returns True	True	☑ PASS
5	MOV-005	Up direction move valid	movement set up, position (10,10)	1. Call is_move_valid((10,10), (9,10), set())	from (10,10) to (9,10)	Returns True	True	☑ PASS
6	MOV-006	Down direction move valid	movement set up, position (10,10)	1. Call is_move_valid((10,10), (11,10), set())	from (10,10) to (11,10)	Returns True	True	☑ PASS
7	MOV-007	Accessible unvisited move valid	movement set up	1. Call is_move_valid to accessible cell	valid parameters	Returns True	True	☑ PASS

3.3 Boundary Violation Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
8	MOV-008	Move outside right boundary fails	env size=20	1. is_move_valid((10,10), (10,20), set())	move to column 20	Returns False	False	☑ PASS
9	MOV-009	Move outside left boundary fails	env size=20	1. is_move_valid((10,0), (10,-1), set())	move to column -1	Returns False	False	☑ PASS
10	MOV-010	Move outside top boundary fails	env size=20	1. is_move_valid((0,10), (-1,10), set())	move to row -1	Returns False	False	☑ PASS
11	MOV-011	Move outside bottom boundary fails	env size=20	1. is_move_valid((19,10), (20,10), set())	move to row 20	Returns False	False	☑ PASS

3.4 Inaccessible Area Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
12	MOV-012	Move to building blocked	env size=100, building at [1:21, 79:99]	1. is_move_valid((10,10), (2,80), set())	move to building	Returns False	False	☑ PASS
13	MOV-013	Move to pond blocked	env size=100, pond at [35:61, 35:61]	1. is_move_valid((34,35), (35,35), set())	move to pond	Returns False	False	☑ PASS
14	MOV-014	Move to border blocked	env size=20	1. is_move_valid((10,10), (10,19), set())	move to border	Returns False	False	☑ PASS

3.5 Visited Position Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
15	MOV-015	Move to visited position blocked	agent visited= (10,10)	1. is_move_valid((11,10), (10,10), visited)	revisit position	Returns False	False	☑ PASS
16	MOV-016	Revisit after other moves blocked	agent visited multiple positions	1. is_move_valid with revisit attempt	attempt revisit	Returns False	False	☑ PASS
17	MOV-017	Multiple revisit attempts all blocked	agent visited positions	1. Attempt revisit multiple times	multiple attempts	All return False	False	☑ PASS
18	MOV-018	Circular path prevention	agent path creates circle	1. Attempt circular move	circular move	Returns False	False	☑ PASS

3.6 Border Detection Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
19	MOV-019	Border detection right edge	agent = Agent((10,10)), env size=20	1. is_move_valid((10,10), (10,20), set()) 2. Check agent.stop called	move right out of bounds	Returns False and agent.stop called	Border detected	☑ PASS
20	MOV-020	Border detection left edge	agent = Agent((10,10)), env size=20	1. is_move_valid((10,0), (10,-1), set()) 2. Check agent.stop called	move left out of bounds	Returns False and agent.stop called	Border detected	☑ PASS
21	MOV-021	Border detection top edge	agent = Agent((10,10)), env size=20	1. is_move_valid((0,10), (-1,10), set()) 2. Check agent.stop called	move up out of bounds	Returns False and agent.stop called	Border detected	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
22	MOV-022	Border detection bottom edge	agent = Agent((10,10)), env size=20	1. is_move_valid((19,10), (20,10), set()) 2. Check agent.stop called	move down out of bounds	Returns False and agent.stop called	Border detected	☑ PASS
23	MOV-023	Border hit stops agent once	agent = Agent((10,10))	1. Hit border twice 2. Check stop_reason not overwritten	two border hits	stop_reason preserved	Preserved	☑ PASS

3.7 Complex Validation Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
24	MOV-024	All conditions pass validation	movement set up correctly	1. is_move_valid with all passing conditions	valid move	Returns True	True	☑ PASS
25	MOV-025	Move fails if any condition fails	movement set up	1. Test with each condition failing	boundary, visited, inaccessible	All return False	False	☑ PASS
26	MOV-026	Corner position inside grid valid	env size=20	1. is_move_valid to (1,1)	corner position	Returns True	True	☑ PASS
27	MOV-027	Corner position at border fails	env size=20	1. is_move_valid to (0,0) or (19,19)	border corner	Returns False	False	☑ PASS
28	MOV-028	Maximum grid position handling	env size=100	1. is_move_valid near (99,99)	max position	Handled correctly	Correct	☑ PASS
29	MOV-029	Diagonal movement application	movement set up	1. is_move_valid with diagonal	diagonal move	Returns True or False as applicable	Correct	☑ PASS
30	MOV-030	Negative coordinate handling	movement set up	1. is_move_valid with negative result	negative coords	Returns False	False	☑ PASS

3.8 State Preservation Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
31	MOV-031	Failed move doesn't update agent position	agent = Agent((10,10))	1. Call is_move_valid with invalid 2. Check position unchanged	invalid move	agent.current_position still (10,10)	Unchanged	☑ PASS
32	MOV-032	Validation consistency	is_move_valid called twice	1. Call is_move_valid with same params twice	same parameters	Both return same result	Same result	☑ PASS
33	MOV-033	Valid move requires manual update	movement set up	1. is_move_valid returns True 2. agent position unchanged	valid move	Position not auto-updated	Unchanged	☑ PASS

3.9 Edge Cases Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
34	MOV-034	Far out of bounds handling	env size=20	1. is_move_valid to (100,100)	very far out	Returns False	False	☑ PASS

Section 4: Action Module Test Cases (50 Tests)

4.1 Initialization Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
1	ACT-001	RIGHT action initialized	Action module imported	1. Check RIGHT action exists	Action class	RIGHT defined	Defined	☑ PASS
2	ACT-002	UP action initialized	Action module imported	1. Check UP action exists	Action class	UP defined	Defined	☑ PASS

4.2 Action Values Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
3	ACT-003	RIGHT value correct	Action module imported	1. Check RIGHT value	RIGHT action	Has specific value	Value verified	☑ PASS
4	ACT-004	LEFT value correct	Action module imported	1. Check LEFT value	LEFT action	Has specific value	Value verified	☑ PASS
5	ACT-005	UP value correct	Action module imported	1. Check UP value	UP action	Has specific value	Value verified	☑ PASS
6	ACT-006	DOWN value correct	Action module imported	1. Check DOWN value	DOWN action	Has specific value	Value verified	☑ PASS

4.3 Delta Calculation Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
7	ACT-007	UP row delta is -1	Action module imported	1. Get UP row delta	UP action	row_delta = -1	Delta verified	☑ PASS
8	ACT-008	DOWN row delta is +1	Action module imported	1. Get DOWN row delta	DOWN action	row_delta = +1	Delta verified	☑ PASS
9	ACT-009	LEFT/RIGHT row delta is 0	Action module imported	1. Get LEFT/RIGHT row deltas	LEFT, RIGHT actions	row_delta = 0	Delta verified	☑ PASS
10	ACT-010	RIGHT col delta is +1	Action module imported	1. Get RIGHT col delta	RIGHT action	col_delta = +1	Delta verified	☑ PASS
11	ACT-011	LEFT col delta is -1	Action module imported	1. Get LEFT col delta	LEFT action	col_delta = -1	Delta verified	☑ PASS
12	ACT-012	UP/DOWN col delta is 0	Action module imported	1. Get UP/DOWN col deltas	UP, DOWN actions	col_delta = 0	Delta verified	☑ PASS
13	ACT-013	All directions have correct deltas	Action module imported	1. Verify all 4 directions	all actions	All deltas correct	All verified	☑ PASS
14	ACT-014	Delta sum magnitude is 1	Action module imported	1. Sum absolute deltas for each	all actions		row_delta	+

4.4 get_all_actions() Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
15	ACT-015	get_all_actions returns 4 actions	Action module imported	1. Call get_all_actions() 2. Check count	none	Returns 4 actions	Count=4	☑ PASS
16	ACT-016	get_all_actions includes all types	Action module imported	1. Call get_all_actions() 2. Check content	none	Returns UP, DOWN, LEFT, RIGHT	All included	☑ PASS

4.5 Action Validation Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
17	ACT-017	RIGHT is valid action	Action module imported	1. Call is_valid_action(RIGHT)	RIGHT action	Returns True	True	☑ PASS
18	ACT-018	Invalid action returns False	Action module imported	1. Call is_valid_action(invalid)	invalid value	Returns False	False	☑ PASS
19	ACT-019	All 4 actions are valid	Action module imported	1. Test each action with is_valid_action	all 4 directions	All return True	True	☑ PASS
20	ACT-020	None is invalid	Action module imported	1. Call is_valid_action(None)	None value	Returns False	False	☑ PASS

4.6 Boundary Value Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
21	ACT-021	Boundary values handled	Action module imported	1. Test min/max action values	boundary values	Handled correctly	Correct	☑ PASS
22	ACT-022	Invalid numeric values rejected	Action module imported	1. Test out-of-range values	invalid numbers	Rejected as invalid	Rejected	☑ PASS

4.7 Opposite Action Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
23	ACT-023	RIGHT opposite is LEFT	Action module imported	1. Call get_opposite(RIGHT)	RIGHT action	Returns LEFT	LEFT returned	☑ PASS
24	ACT-024	UP opposite is DOWN	Action module imported	1. Call get_opposite(UP)	UP action	Returns DOWN	DOWN returned	☑ PASS
25	ACT-025	Opposite function consistency	Action module imported	1. Test opposite(opposite(a))	all actions	Returns original action	Original returned	☑ PASS
26	ACT-026	All opposites work	Action module imported	1. Test all 4 directions	all actions	All have correct opposites	All correct	☑ PASS

4.8 Movement Application Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
27	ACT-027	RIGHT moves to (r,c+1)	Action module imported	1. Apply RIGHT to (5,5)	position=(5,5), action=RIGHT	Result = (5,6)	(5,6) returned	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
28	ACT-028	LEFT moves to (r,c-1)	Action module imported	1. Apply LEFT to (5,5)	position=(5,5), action=LEFT	Result = (5,4)	(5,4) returned	☑ PASS
29	ACT-029	UP moves to (r-1,c)	Action module imported	1. Apply UP to (5,5)	position=(5,5), action=UP	Result = (4,5)	(4,5) returned	☑ PASS
30	ACT-030	DOWN moves to (r+1,c)	Action module imported	1. Apply DOWN to (5,5)	position=(5,5), action=DOWN	Result = (6,5)	(6,5) returned	☑ PASS
31	ACT-031	Apply action to position works	Action module imported	1. Apply any action to position	generic application	Result calculated correctly	Correct	☑ PASS
32	ACT-032	Sequential actions applied correctly	Action module imported	1. Apply multiple actions sequentially	sequence of actions	Final position correct	Correct	☑ PASS

4.9 Consistency Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
33	ACT-033	RIGHT value constant	Action module imported	1. Check RIGHT multiple times	RIGHT action	Value unchanged	Constant	☑ PASS
34	ACT-034	LEFT value constant	Action module imported	1. Check LEFT multiple times	LEFT action	Value unchanged	Constant	☑ PASS
35	ACT-035	UP value constant	Action module imported	1. Check UP multiple times	UP action	Value unchanged	Constant	☑ PASS
36	ACT-036	DOWN value constant	Action module imported	1. Check DOWN multiple times	DOWN action	Value unchanged	Constant	☑ PASS
37	ACT-037	Delta values consistent	Action module imported	1. Check deltas multiple times	all actions	Deltas unchanged	Consistent	☑ PASS
38	ACT-038	Action names consistent	Action module imported	1. Check names multiple times	all actions	Names unchanged	Consistent	☑ PASS
39	ACT-039	All action names unique	Action module imported	1. Compare all action names	all actions	No duplicates	All unique	☑ PASS
40	ACT-040	All action values unique	Action module imported	1. Compare all action values	all actions	No duplicates	All unique	☑ PASS

4.10 Edge Case Tests

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
41	ACT-041	Action on (0,0) position	Action module imported	1. Apply each action to (0,0)	position=(0,0)	Works correctly	Correct	☑ PASS
42	ACT-042	Repeated same action	Action module imported	1. Apply same action multiple times	repeated action	Accumulates correctly	Correct	☑ PASS

No.	Test Case ID	Test Case Objective	Prerequisite	Steps	Input Data	Expected Result	Actual Result	Status
43	ACT-043	Four opposite actions return to origin	Action module imported	1. Apply RIGHT 2. LEFT 3. UP 4. DOWN	sequence of opposites	Returns to (5,5)	Origin reached	☑ PASS
44	ACT-044	Boundary positions handled	Action module imported	1. Apply actions near boundaries	boundary positions	Handled correctly	Correct	☑ PASS
45	ACT-045	Negative movement handled	Action module imported	1. Apply actions causing negative	negative coordinates	Negative coords possible	Possible	☑ PASS
46	ACT-046	Actions are comparable	Action module imported	1. Compare actions for equality	action comparison	Equality works	Works	☑ PASS
47	ACT-047	Action string representation	Action module imported	1. Convert action to string	string conversion	Readable format	Format valid	☑ PASS
48	ACT-048	Actions are iterable	Action module imported	1. Iterate all_actions	iteration	Can iterate all	Iterable	☑ PASS
49	ACT-049	Large position coordinates	Action module imported	1. Apply action to large coords	large position	Works with large numbers	Works	☑ PASS
50	ACT-050	get_all_actions returns list	Action module imported	1. Check return type	none	Type is list	List returned	☑ PASS

Test Summary

Category	Total Tests	Passing	Failing	Pass Rate
Environment	43	43	0	100%
Agent	46	46	0	100%
Movement	34	34	0	100%
Action	50	50	0	100%
TOTAL	**173**	**173**	**0**	**100%**

Test Execution Instructions

Run All Tests

```
python -m pytest tests/ -v
```

Run by Module

```
python -m pytest tests/test_environment.py -v
python -m pytest tests/test_agent.py -v
python -m pytest tests/test_movement.py -v
python -m pytest tests/test_action.py -v
```

Run Specific Test

```
python -m pytest tests/test_module.py::TestClass::test_method -v
```

Generate Coverage Report

```
python -m pytest tests/ --cov=. --cov-report=html
```

Document Status: ☑ COMPLETE
All Tests Passing: ☑ YES
Ready for Production: ☑ YES