Requirement ID	Description of Requirement	Contributer Name	Story Point	Sprint Number
1	Creation of Story Board	Brandon Dodge	1	1
2	Create GitHub repository, setup branching strategy, and initial README	Zach Sevart	2	1
3	Add person-hours log template and contribution guidelines	Josh Dwoskin	2	1
4	Draft architecture diagram (Board Manager, Game Logic, UI, Input Handler)	Brandon Dodge	3	1
5	Write /docs/architecture.md with system description	Brandon Dodge	3	1
6	Implement 10×10 grid with cell states	Asa Maker	3	1
7	Implement mine placement logic (10–20 mines randomly placed)	Asa Maker	5	1
8	Implement uncover cell logic (including number calculation for adjacent mines)	Zach Sevart	8	1
9	Implement flag/unflag cells with flag count update	Asa Maker	3	1
10	Prologue comments in all files (name, inputs/outputs, author, attribution)	Ebraheem	2	1
11	Inline comments for Board Manager and Game Logic modules	Ebraheem	2	1
12	First Revisions for System Architecture	Josh Dwoskin	2	2
13	Second Draft for System Architecture	Brandon Dodge	3	2
14	Project Overview Checklist	Brandon Dodge	3	2
15	As a player, when I click a cell with zero adjacent mines, all adjacent cells are automatically uncovered recursively/Flood Fill	Zach Sevart	8	2
16	As a player, the game should detect when I have uncovered all non-mine cells and declare victory.	Asa Maker	5	2
17	UML Case Diagram	Brandon Dodge	3	2
18	As a player, the game should detect when I click on a mine, end the game, and declare a loss.	Asa Maker	3	2
19	Visual indicator for number of mines	Zach Sevart	3	2
20	10x10 grid that renders the state of each cell	Zach Sevart	3	2
21	Class Diagram	Josh Dwoskin	3	2
22	System Context Diagram	Josh Dwoskin	3	2

23	Basic unit testing	Asa Maker	2	2

Name	# Total hours	Tт Contributions	Date
Brandon Dodge	3	Designed the initial high-level architecture for PySweeper, outlining core components such as the Board Manager, Game Logic, User Interface, and Input Handler. This provided the team with a foundation to build consistent and extensible features.	9/8/25
Brandon Dodge	1	Created the first iteration of the use case diagram, mapping interactions between the player and the system (e.g., uncovering cells, flagging mines, and game-ending conditions). This clarified system requirements and user workflows.	9/8
Josh Dwoskin	3	created the class diagram for our project, which shows the main classes, their attributes, methods, and relationships. This helped the team visualize the system structure before coding and made it easier to spot dependencies and plan responsibilities.	9/15
Josh Dwoskin	2	created the system context diagram, which illustrates how our software interacts with external entities such as users, databases, and other systems. This gave the team a clear high-level view of the system boundaries and main inputs/outputs.	9/15
Brandon Dodge	1.5	Reviewed peer feedback and refined the architecture documentation to fix inconsistencies, improve clarity, and align with updated requirements.	9/13/25
Team	1	Collaboratively developed the overall project plan, including milestones, deliverables, and role assignments to keep the team on track.	9/8/25
Team	1	Group play-testing and debugging of the PySweeper prototype, identifying edge cases (e.g., recursive uncovering, flag miscounts) and refining gameplay.	9/17/25
Brandon Dodge	3	Compiled, polished, and finalized the architecture documentation, ensuring it met course requirements and was submission-ready.	9/19/25
Asa Maker	1.5	Core Architecture & Scaffolding: This task was completed exactly on schedule. The Cell and Board classes were designed and implemented as planned, including a beneficial refactor to store screen coordinates within each Cell object.	9/8
Asa Maker	0.75	Adjacent Mine Calculation: This standard algorithm was implemented without issue and met its time estimate precisely.	9/9
Asa Maker	6	Recursive Reveal Algorithm & Debugging: This single task was the source of the project's time variance. Initial Implementation (2.0 hrs): The first version of the function was coded within the estimated timeframe. Bug Fixing (4.0 hrs): Extensive testing revealed a deep-seated logical flaw that caused an intermittent but critical RecursionError on complex grids. This unplanned debugging effort required 4.0 hours of intensive tracing and multiple approaches to finally isolate and resolve the issue, exceeding the entire initial buffer for this task.	9/10-9/12
Asa Maker	1.25	"Safe First Click" Logic: This feature took slightly longer than estimated. To ensure correctness, I implemented a more robust solution that generates a list of valid mine locations after removing a safe zone, rather than a simpler "move-if-hit" approach.	9/12
Asa Maker	1	Final Logic Integration & Review: Integration with the UI revealed a minor state management bug related to the game reset function. This, along with a comprehensive final playtest, accounted for the additional half hour.	9/13
Ebraheem AlAamer	2	Finished and completed all the needed comments for the codes and commented on each code to make sure it's perfect and clear	9/21
Zach Sevart	2	Implemented cell uncover logic on first click (start of game and mine generation)	9/13

Name	# Total hours	Tт Contributions	Date
Zach Sevart	3	Created UI to match original Minesweeper UI, added timer to track individual game times, gathered and formatted sprites to use for UI	9/18
Zach Sevart	1	GitHub maintenence, reviewed and approved all commits to ensure main branch was preserved (main project)	9/8 - 9/21

Name	Total hours	Contributions	Methodology	Date
Brandon Dodge	3	System Architecture Form	Define the overall structure of the system to clearly outline components and their interactions.	9/8/25
Brandon Dodge	1	Use Case Diagram	Illustrate user interactions with the system in a clear, visual format.	9/8
Josh Dwoskin	2	Class Diagram	Map out the system's classes and relationships to clarify design.	9/15
Josh Dwoskin	1	System Context Diagram	Show how the system fits into its external environment and connects with outside elements.	9/15
Brandon Dodge	2	System Architecture Corrections	Refine and adjust the design to address gaps or inconsistencies.	9/13/25
Team	1	Project Plan	Organize tasks and timelines to guide development effectively.	9/8/25
Team	1	Product Testing	Validate functionality through systematic testing and feedback cycles.	9/17/25
Ebraheem AlAamer	2	Code comments	Improve code readability by explaining logic and intent where needed.	9/21/25
Asa Maker	1.5	Core Logic Architecture & Scaffolding: I will begin by designing and implementing the Cell and Board classes. This involves defining the necessary state variables and creating the foundational 2D grid structure for the game.	Build the foundational logic framework to support game functionality.	9/8
Asa Maker	0.75	Adjacent Mine Calculation: I will implement the logic to iterate through the grid and calculate the number of adjacent mines for each non-mine cell. This is a standard grid algorithm and should be straightforward.	Implement logic to determine mine counts around each cell.	9/9
Asa Maker	3	Recursive Cell Reveal Algorithm (Initial Implementation): This is the most complex planned task. I will develop the recursive "flood fill" function that reveals empty areas. This is the primary risk area for my work, as debugging the base cases for recursion can be time-intensive.	Develop an algorithm to uncover large empty areas of the board efficiently.	9/9
Asa Maker	0.75	"Safe First Click" Logic: As a quality-of-life feature, I will implement logic to ensure the player's first click is never a mine.	Ensure the first player move always results in a playable start.	9/11
Asa Maker	0.5	Final Logic Integration & Review: I will allocate a short period for integrating my completed logic with the Pygame front end and performing a final functionality check.	Combine and verify all logic components for seamless operation.	9/14
Zach Sevart	2	UI	Design and implement a user interface that is clear and responsive.	9/14
Zach Sevart	3	Recursive Cell reveal on start	Integrate the reveal algorithm to enhance gameplay experience from the beginning.	9/14
Brandon Dodge	3	System Architecture Form Finalizations	Finalize and document the system's design for consistency and future reference.	9/19