InternPro Weekly Progress Update

Name	Email	Project Name	NDA/ Non-NDA	InternPro Start Date	OPT
Adharsh Prasad Natesan	anatesan@asu.edu	IT-Core Foundation Suriname	Non-NDA	2024-08-05	Yes

Progress

Include an itemized list of the tasks you completed this week.

#	Action Item/ Explanation	Total Time This Week (hours)
1	Particle Class Initialization	2
2	Rover Class Definition	2
3	Create a Unified Environment with Particles and a Rover	3
4	Create a 3D environment that integrates a moving rover and multiple static particles	3
5	Efficient large quantity moving particle simulation	2
6	Research Paper on Creating a Tilling/Ploughing Rover in Python	2
7	Next week plan	1
8	Report writing	1
9	Holiday entry	2
10	Holiday entry	2
	Total hours for the week:	20

Verification Documentation:

Action Item 1: Particle Class Initialization – 2 hour(s).

Project Work Summary

- Initialization
  - Define initial position, velocity (which can be zero if the particle starts at rest), radius (or size), and mass.

    - Optionally provide a random velocity generator or a function for external force assignment.
- Force Calculation
    - Create a method (e.g., calculate_force) that determines the net force on each particle.
    - Include gravity (a constant downward force) and possibly other interactions such as collisions with walls or the rover.
- Update Mechanics
    - Define a function (e.g., update_particle) that incorporates forces into the velocity and updates the position.
    - Implement collision detection with boundaries. If a collision occurs, invert or modify velocity to simulate a bounce and apply a damping factor.
- Visualization
    - Although the class itself may not directly visualize, it will store the properties needed (position, velocity) for rendering in an environment such as Matplotlib or Mayavi.
    - If using 2D, represent each particle with a colored circle.
    - If using 3D, represent each particle with a small sphere or point symbol at the correct coordinates.

Action Item 2: Rover Class Definition – 2 hour(s).

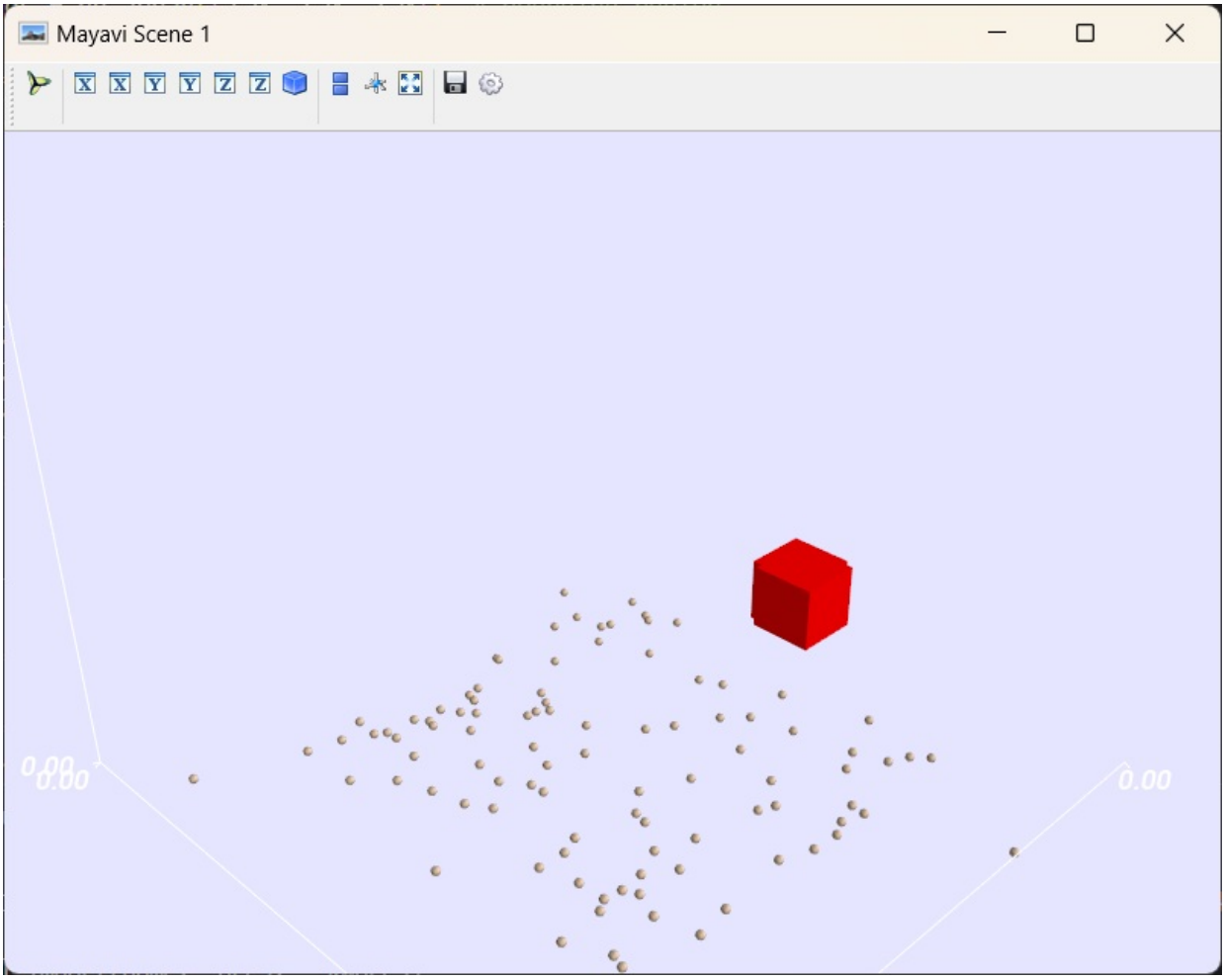
Project Work Summary

- Rover Class Definition
    - Store the rover’s center position, size (length, width, height), and velocity.
    - For realism, you can subdivide the rover into wheels, chassis, and mass distribution, but a simple cuboid is sufficient for a basic simulation.
- Movement
    - Implement a function like update_rover that moves the rover according to its velocity.
    - Detect boundary limits if the rover is constrained to a certain domain, applying any necessary stops or reflections.
- Interaction with Particles
    - Provide a method to check for collisions or proximity between the rover’s bounding box and nearby particles.
    - When collisions occur, apply a contact force or impulse to the particle, causing it to gain velocity and move away.
- Visualization
    - Similar to the Particle class, store attributes (e.g., bounding box corners) that the graphics layer can read for rendering.
    - In 2D, visualize it as a rectangle. In 3D, plot a cuboid (using lines, points, or a solid shape) corresponding to the rover dimensions.

Action Item 3: Create a Unified Environment with Particles and a Rover – 3 hour(s).

Project Work Summary

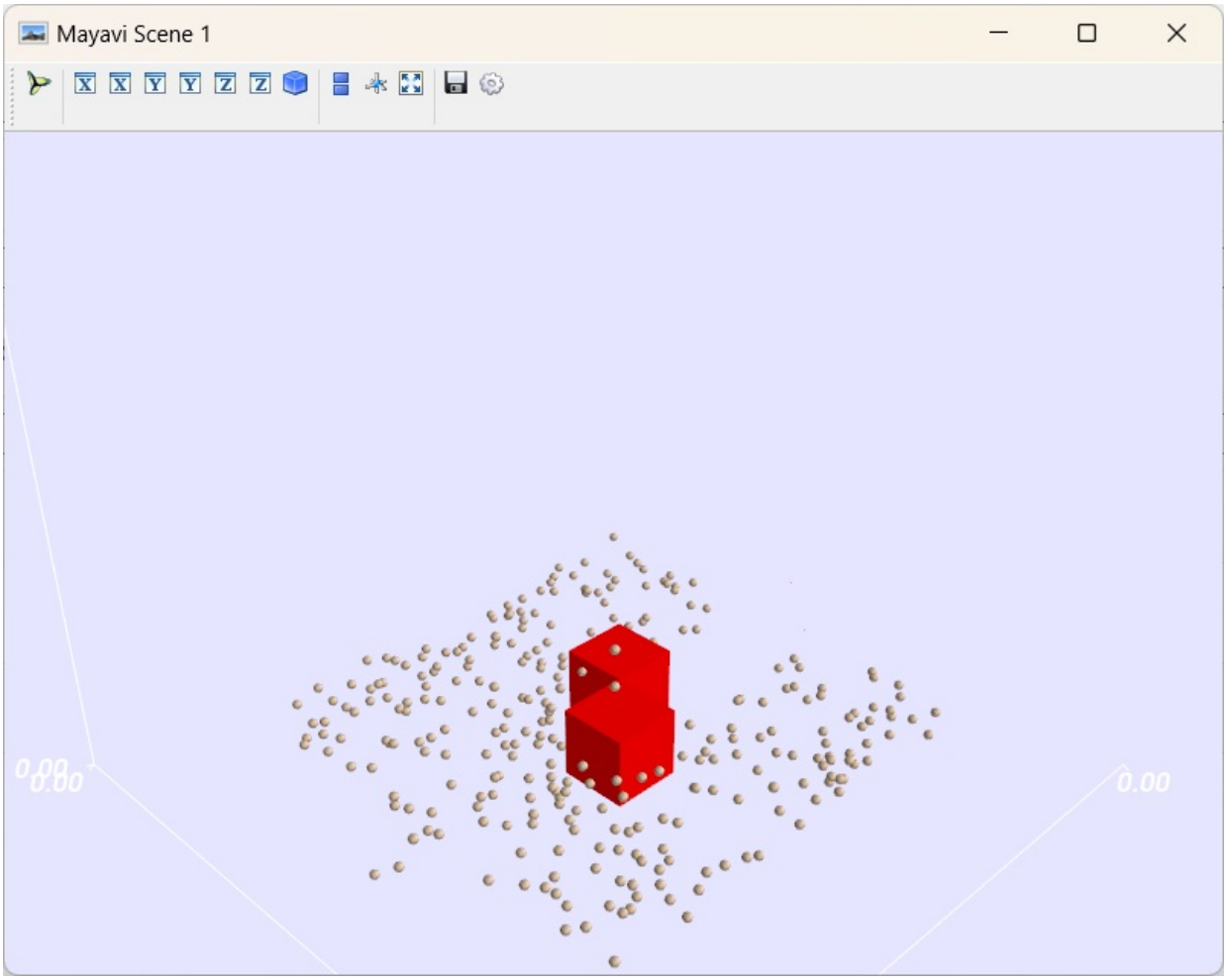
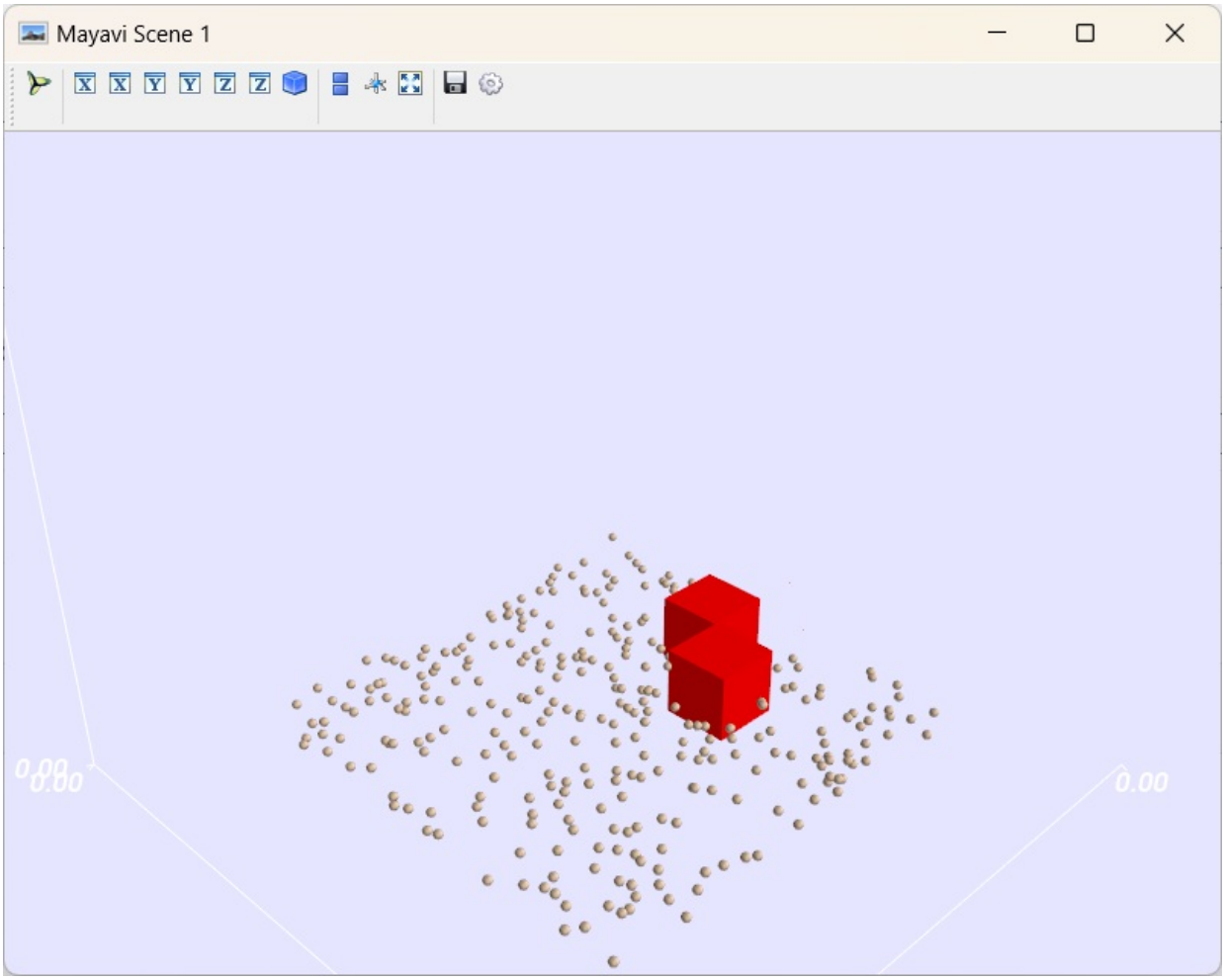
- Environment Setup
  - Generate or define a 3D domain to house both the rover entity and the sand-like particles.
  - Use a regular grid or a pre-computed terrain surface if desired, ensuring enough space for future rover navigation.
  - Establish boundary conditions (e.g., walls or edges) that constrain both particles and the rover’s position within the simulation[1].
- Particle Integration
  - Instantiate a set of Particle objects that will occupy the environment.
  - Store each particle’s position, radius, and mass; keep initial velocities at zero to have them at rest until interaction begins.
  - (Optional) Include basic granular properties, so the particles can be easily extended to support collisions, friction, or cohesion.
- Rover Placement
  - Instantiate a Rover object with a defined center point, size (length, width, height), and zero velocity for now.
  - Compute a bounding box from the rover’s dimensions for collision checks.
  - Position the rover so that it sits on or near the particle layer, ready for future movement and interaction.
- Visualization Preparations
  - Include a 3D visualization module (e.g., Mayavi) to render the environment.
  - Plot particles at their initial positions with a suitable scale factor, color, and shape.
  - Render the rover’s bounding box or a cuboid representation in a contrasting color to distinguish it from the particles.

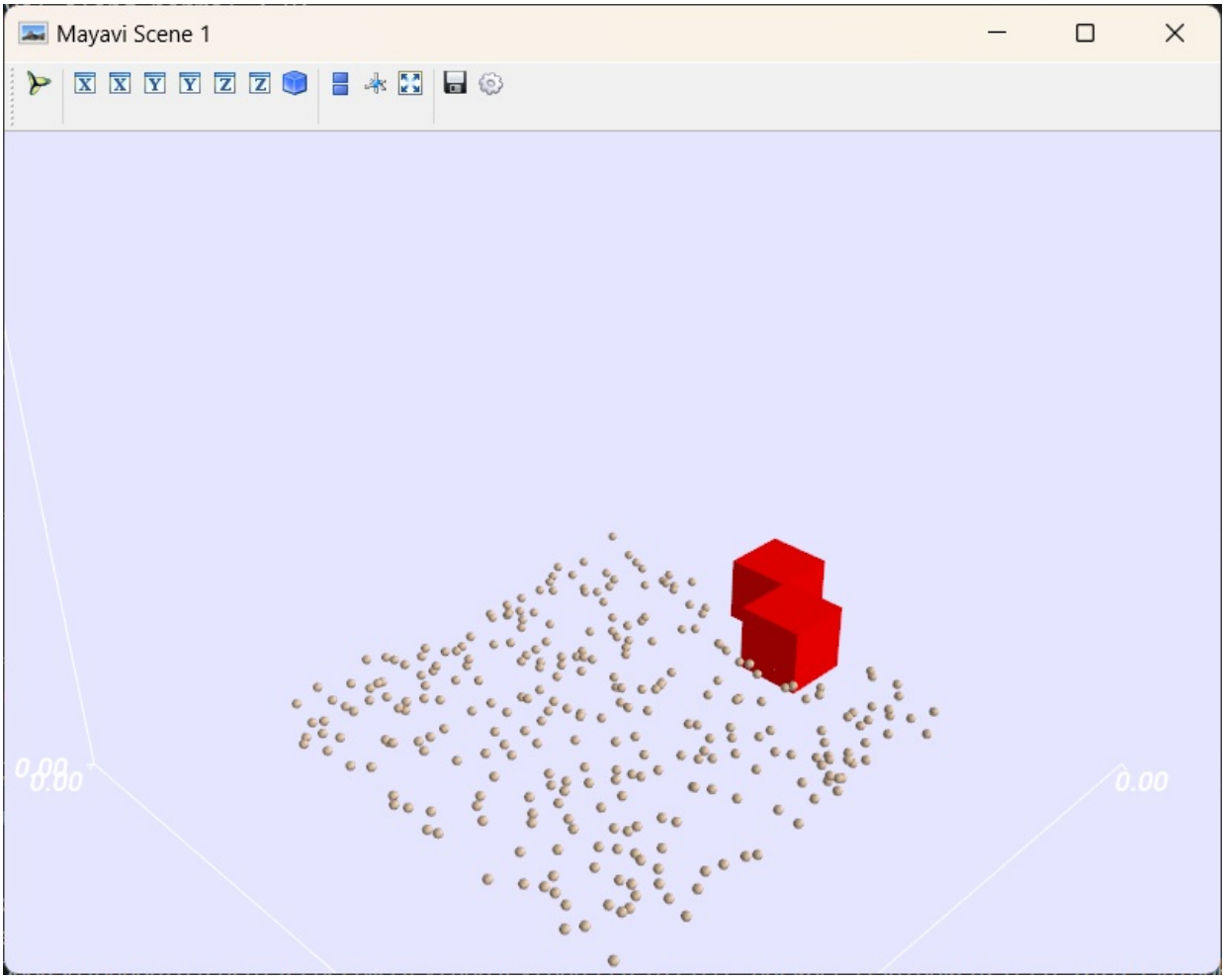


Action Item 4: Create a 3D environment that integrates a moving rover and multiple static particles – 3 hour(s).

Project Work Summary

- Environment Setup
  - Define a 3D space (e.g., 5×5×2 meters) where particles and rover coexist.
  - Initialize particles in their positions at rest with zero velocity. These particles remain static unless acted upon by the rover.
- Particle Class
  - Store each particle's attributes (position, radius, mass), keeping them non-moving by default.
  - Include collision methods so particles can be pushed by the rover and interact with neighboring particles if they gain velocity.
- Rover Class
  - Represent the rover as a cuboid with a center coordinate, size (length, width, height), and a specified velocity.
  - Implement a function to compute its bounding box and detect overlaps with particles.
  - When the bounding box overlaps a particle, add a fraction of the rover's velocity to the particle, causing it to move.
- Collision Handling
  - Check particle-particle overlap to propagate motion if an impacted particle collides with adjacent ones.
  - Add boundary checks to keep objects in the simulation domain.
- 3D Visualization
  - Render particles as small spheres or points in a consistent color (e.g., sand-colored).
  - Visualize the rover's bounding box in a contrasting color for clarity.
  - Continuously update positions in real time, allowing the rover to move while particles stay static until pushed.
- Extensions
  - Incorporate additional forces like friction or damping if particles should settle or slow after being displaced.
  - Introduce gravity selectively if required (though in this setup, particles remain at rest until collision)
- This approach draws upon earlier steps for multi-particle collision and 3D rover insertion, ensuring static particles that move only when contacted by the rover's bounding box or by other particles set in motion[1].

- 
Mayavi Scene 1
0.00
0.00
- 
Mayavi Scene 1
0.00
0.00



Action Item 5: Efficient large quantity moving particle simulation – 2 hour(s).

Project Work Summary

- http://mdpi.com/2673-3951/5/1/15
- An Efficient Explicit Moving Particle Simulation Solver for Simulating Free Surface Flow on Multicore CPU/GPUs
- Summary of Report
  - Proposes an explicit solver tailored to handle free-surface flow with a moving particle method.
  - Uses adaptive neighbor-finding strategies to reduce the overhead of large-scale computations.
  - Achieves real-time or near-real-time performance for dam-break simulations with thousands of particles.
- Relation to Project
  - Demonstrates how specialized solvers (e.g., MPS or SPH) can be optimized for fluid-like particle motion in HPC contexts.
  - Underlines the utility of an explicit time-integration scheme, simplifying parallelization.
  - Confirms that careful data structures (e.g., hash grids or cell-based indexing) can enhance performance at scale.
- Motivation for Research
  - Improve the speed of particle-based free-surface simulations without compromising numerical stability.
  - Provide a foundation for coupling large-scale hydrodynamic flows with granular or solid particles.
  - Facilitate efficient fluid-structure interaction studies in research and engineering applications.

Action Item 6: Research Paper on Creating a Tilling/Ploughing Rover in Python – 2 hour(s).

Research

- https://www.researchgate.net/publication/317308250_Agricultural_robot_designed_for_seeding_mechanism
- Agricultural Robot Designed for Seeding Mechanism

- Summary of Report
  - Introduces a ground-based robot equipped with plough-like attachments capable of basic tilling and seed placement.
  - Discusses a modular architecture: sensors, actuators, and control logic working together to automate ground preparation.
  - Outlines an approach that can be adapted using Python for controlling motors, reading sensor data, and executing path planning in agricultural fields.
- Relation to Project
  - Serves as a direct reference for implementing rover-based tillage in Python, using straightforward control scripts for traction and tool actuation.
  - Provides design ideas for a lightweight robotic chassis and the attachments needed for ploughing, seeding, or other field tasks.
  - Offers a prototype that can be further developed or integrated into larger precision-agriculture frameworks.
- Motivation for Research
  - Reduce the manual labor required for soil preparation and seeding, especially in small to medium-sized farms.
  - Enhance the precision of seed placement and consistent soil turning using a programmable rover.
  - Lay groundwork for more robust autonomy (e.g., sensor fusion, navigation algorithms) that can be coded in Python for accessibility and expandability.

Action Item 7: Next week plan – 1 hour(s).

Project Work Summary

- Extend 3D Terrain Layering
  - Introduce multiple soil strata with different properties (e.g., density, friction angle).
  - Load these strata from separate data files or parameter sets, then fill the terrain volume accordingly for better realism.
  - Include a configuration option to toggle specific soil types on or off for testing different scenarios.
- Particle-Particle Collision Optimization
  - Replace the brute-force collision loop with a cell- or grid-based method to reduce computational overhead.
  - Add a tunable friction or rolling-resistance parameter so that post-collision particle velocities diminish naturally over time.
  - Verify performance improvements by measuring frame rate or simulation step duration under different particle counts.
- Enhanced 2D Visualization Tools
  - Combine the existing multiple-particle 2D simulations into one interface where parameters (particle count, collision elasticity, damping) can be changed on the fly.
  - Add a simple real-time chart or textual readout (e.g., total kinetic energy of system) to quantify particle dynamics.
  - Experiment with adding additional “objects” or geometric shapes that can collide with particles for more varied 2D scenarios.

Action Item 8: Report writing – 1 hour(s).

Project Work Summary

- Created word document layout to write contents of the weekly progress.
- Created relevant subsections in the epicspro website and documented 20 hours of weekly progress.
- Collected relevant documents research papers, relevant links and company's objective from their portal.

Action Item 9: Holiday entry – 2 hour(s).

ASU holiday and corresponding hours entry to complete the 20 hours per week, so that I could submit my progress.

Action Item 10: Holiday entry – 2 hour(s).

ASU holiday and corresponding hours entry to complete the 20 hours per week, so that I could submit my progress.