Howitzer Simulator

Ahras Ali, Bulbul Arora, Jasmeet Singh

Our Team

Ahras Ali

Bulbul Arora

Jasmeet Singh

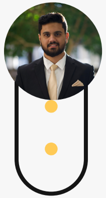
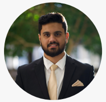


Table of **contents**



Fun fact

The word "howitzer" is originated from the Czech term "houfnice," which means "a swarm of wasps." This name was given because of the buzzing sound the projectile made as it flew in the air.

01

Introduction

Introduction/Approach

Background:

This project focuses on the development of a Howitzer simulator application, a significant tool for military training and operations. The application, written in Java, employs rigorous testing strategies to ensure accuracy and reliability.

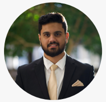
Rationale:

The simulator is designed with a user-centric approach. We prioritized usability over system orientation, focusing on an intuitive and engaging experience. Instead of waiting for implementation after multiple MVPs, our goal was to create a tool ready for deployment with essential features right from the start.

Testing Strategy:

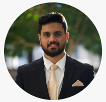
Our comprehensive testing approach ensures reliability and robustness, utilizing various strategies:

- Node Coverage with Data Flow
- Integration testing
- Decision table-based testing
- Boundary value testing
- Equivalence class testing
- System testing via Finite State Machines



02

Design Requirements



Design Requirements

Objectives:

The objective of this project is to create test suites for a Howitzer Firing Simulator. The main objective is not to design the simulator but to design tests in order to verify the kinematic and dynamic of the simulator.

Functions:

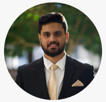
The functions of the Howitzer must include loading the projetial, launching the projetical and analyzing the results

Testing:

The project must include Integration testing and system testing
The project must include testing paths and node coverage.

Code Management:

The code must be managed on a cloud environment and accessible to all members of the team.



Design Constraints

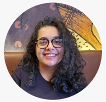
Economic Factors:
The project will consider the cost of the equipment, the cost of implementation, operational costs and maintenance costs. Furthermore, The project will assess ways of cost reduction for long term deployment.

Regulatory Compliance (Security and Access):
The simulator should abide by the security standards and the protocols to protect sensitive information such as encryption of the data and access control measures.

Ethics:
The main ethical consideration when designing the simulator is not having any harmful or malicious intentions and it should strictly follow the training rules when in use. Data privacy should be handled in a secure manner and following regulatory data protection laws.

Societal Impacts:
The design of the stimulator should be usable by all groups of people, meaning that the design should be user friendly and environmentally friendly. The main considerations would be interface and accommodating individuals with disabilities.

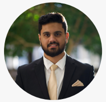
Testing Boundaries



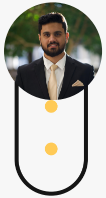
Variables	Nominal
Barrel Pose (x, y, z)	0,0,0
Radius	0 > radius <= 0.25
Mass	0 > mass <= 50
Initial Speed	10
External Force (x, y, z)	100, 100, 0

03

Solution



Solution



MVP 1: Basic Functionality
- Load the simulator
- Set the barrel position
- Fire the simulator
- Calculate the trajectory

Note: MVP 1 is functional, but lacks safety measures.

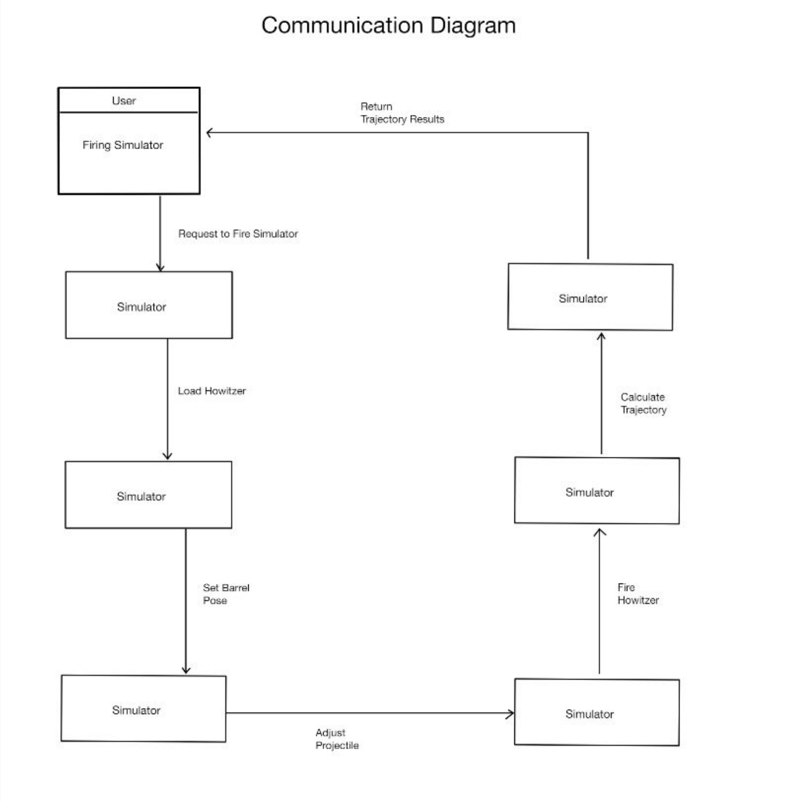
MVP 2: Testing and Safety Integration
- Integrate safety measures using guards and triggers
- Implement testing strategies:
  - Unit Testing
  - Boundary Value Testing
  - Decision table-based testing

Note: MVP 2 aims to ensure the simulator meets project criteria by enhancing functionality and safety.

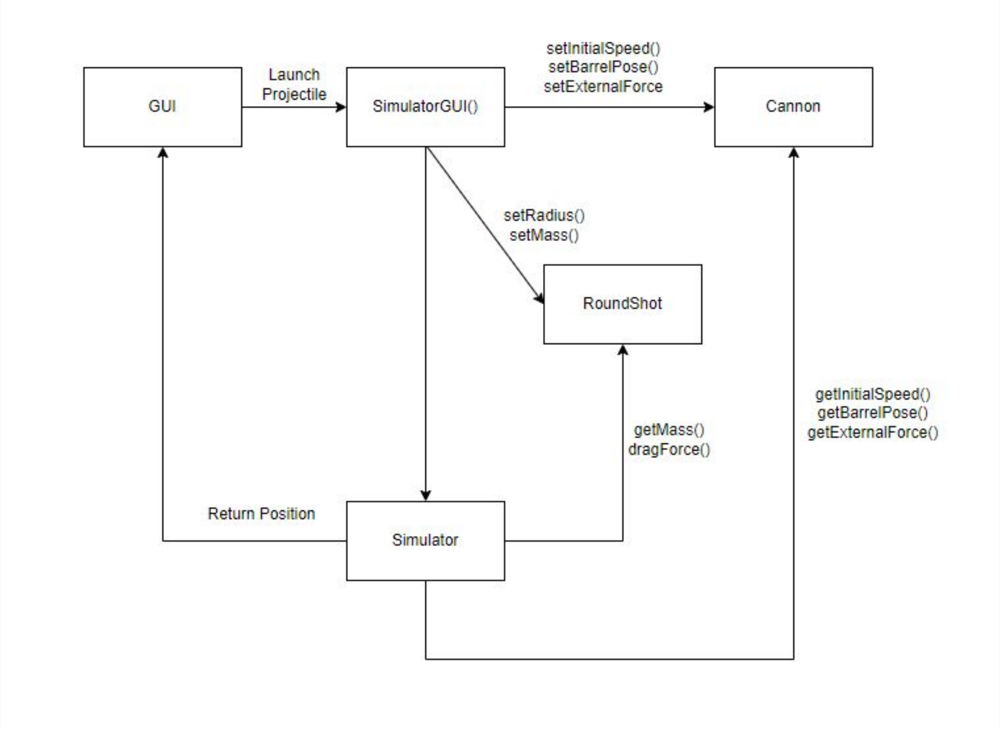
MVP 3: Enhanced Features and Design
- Calculate maximum height reached by projectile
- Implement an object-oriented design with different classes
- Additional enhancements building upon MVP 2

Note: This stage focuses on expanding features and refining the system design.

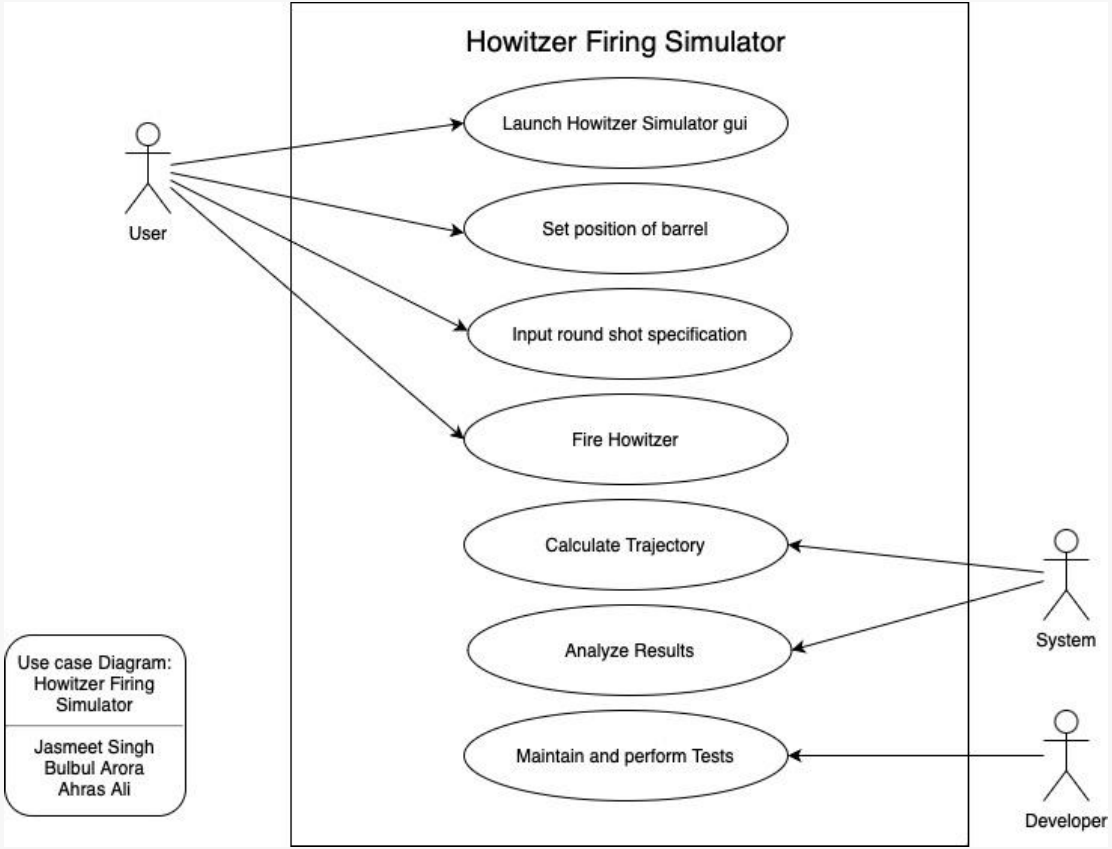
Communication Diagram Version 1



Communication Diagram



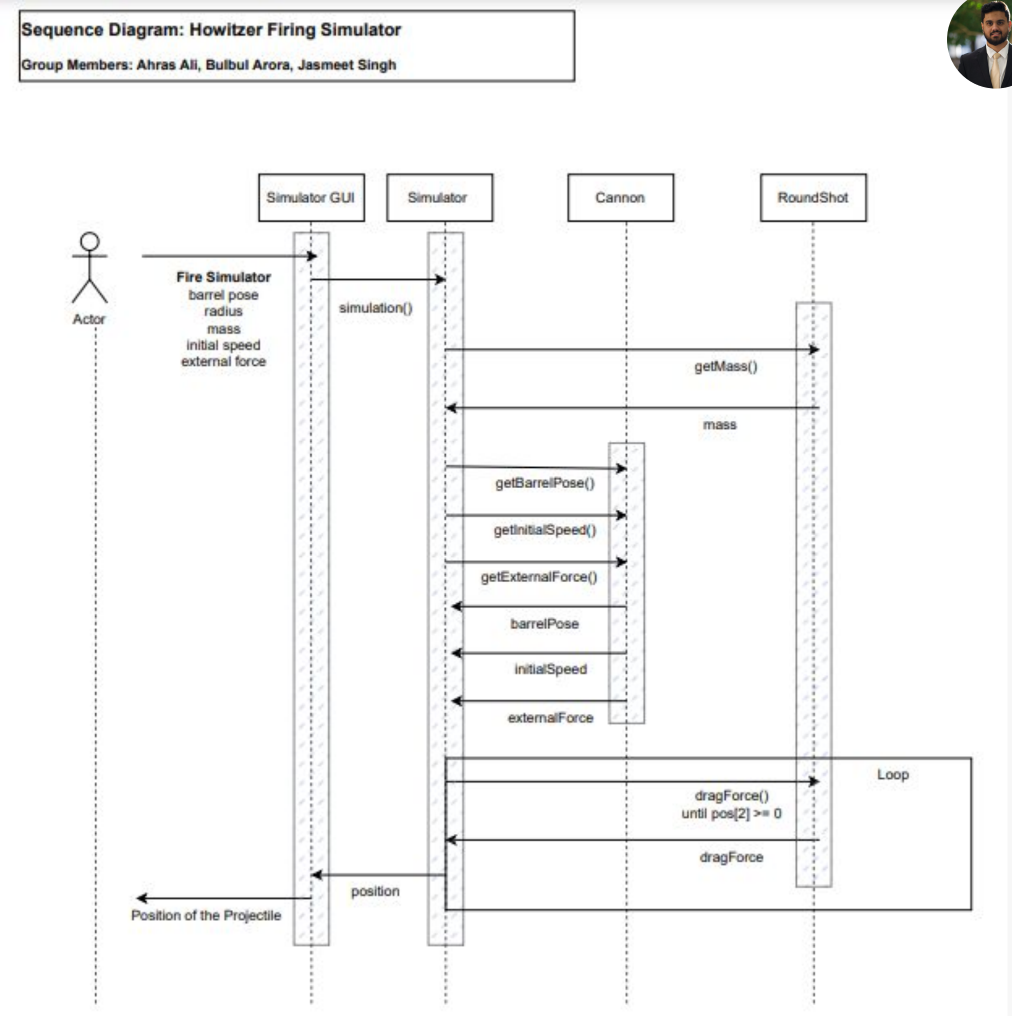
Use Case Diagram



Sequence Diagram

Our sequence diagram is a diagram that describes the interactions between different components of our project such as the gui, simulator, canon and RoundShot.

The main focus of our sequence diagram is the loop which runs until the z component is zero which mean the round shot has impacted the ground. In this loop, the simulator talks to the round shot class to get the new drag force for each iteration of the loop.



04

Testing

Boundary Value Testing (Version 1 and 2)

Boundary Value Testing

Testing to see if we get null or non- null values. Initial Boundary value Testing with very exaggerated boundaries.

Table 1: Variables and Values Tested

Variables	Minimum	Maximum	Nominal
Barrel Pose (x, y, z)	Negative Infinity	Positive Infinity	0
Radius	0	Positive Infinity	0 > radius < 0.25
Mass	0	Positive Infinity	0 > mass < 10
Drag Coefficient	0	Positive Infinity	0.5
Initial Speed	0	Positive Infinity	10
External Force (x, y, z)	Negative Infinity	Positive Infinity	100

Boundary Value Testing (Version 1 and 2)

Table 2: Results for Initial Boundary Value Testing (Exaggerated Values)

Case	Barrel Pose (x, y, z)	Radius	Mass	Drag Coefficient	Initial Speed	External Force (x, y, z)	Results
1	**Negative Infinity**	Nominal	Nominal	Nominal	Nominal	Nominal	Not Null
2	**0**	Nominal	Nominal	Nominal	Nominal	Nominal	Not Null
3	**Positive Infinity**	Nominal	Nominal	Nominal	Nominal	Nominal	Not Null
4	Nominal	**0**	Nominal	Nominal	Nominal	Nominal	Null
5	Nominal	**0.25**	Nominal	Nominal	Nominal	Nominal	Not Null
6	Nominal	**Positive Infinity**	Nominal	Nominal	Nominal	Nominal	Not Null
7	Nominal	Nominal	**0**	Nominal	Nominal	Nominal	Null
8	Nominal	Nominal	**1**	Nominal	Nominal	Nominal	Not Null
9	Nominal	Nominal	**Positive Infinity**	Nominal	Nominal	Nominal	Not Null
10	Nominal	Nominal	Nominal	**0**	Nominal	Nominal	Null
11	Nominal	Nominal	Nominal	**0.5**	Nominal	Nominal	Not Null
12	Nominal	Nominal	Nominal	**Positive Infinity**	Nominal	Nominal	Not Null
13	Nominal	Nominal	Nominal	Nominal	**0**	Nominal	Null
14	Nominal	Nominal	Nominal	Nominal	**10**	Nominal	Not Null
15	Nominal	Nominal	Nominal	Nominal	**Positive Infinity**	Nominal	Not Null
16	Nominal	Nominal	Nominal	Nominal	Nominal	**Negative Infinity**	Not Null
17	Nominal	Nominal	Nominal	Nominal	Nominal	**100**	Not Null
18	Nominal	Nominal	Nominal	Nominal	Nominal	**Positive Infinity**	Not Null

Table 3: Results for Initial Boundary Value Testing (With Guards on Mass, radius and Drag Coefficient)

Case	Barrel Pose (x, y, z)	Radius	Mass	Drag Coefficient	Initial Speed	External Force (x, y, z)	Expected and tested Results (Position – x, y, z)
1	**Negative Infinity**	Nominal	Nominal	Nominal	Nominal	Nominal	Not Null
2	**0**	Nominal	Nominal	Nominal	Nominal	Nominal	Not Null
3	**Positive Infinity**	Nominal	Nominal	Nominal	Nominal	Nominal	Not Null
4	Nominal	**0**	Nominal	Nominal	Nominal	Nominal	Throws Illegal Argument Exception
5	Nominal	**0.25**	Nominal	Nominal	Nominal	Nominal	Not Null
6	Nominal	**1**	Nominal	Nominal	Nominal	Nominal	Throws Illegal Argument Exception
7	Nominal	Nominal	**0**	Nominal	Nominal	Nominal	Throws Illegal Argument exception
8	Nominal	Nominal	**1**	Nominal	Nominal	Nominal	Not Null
9	Nominal	Nominal	**10**	Nominal	Nominal	Nominal	Not Null
10	Nominal	Nominal	Nominal	**0**	Nominal	Nominal	Throws Illegal Argument Exception
11	Nominal	Nominal	Nominal	**0.5**	Nominal	Nominal	Not Null
12	Nominal	Nominal	Nominal	**1**	Nominal	Nominal	Throws Illegal Argument Exception
13	Nominal	Nominal	Nominal	Nominal	**0**	Nominal	Null
14	Nominal	Nominal	Nominal	Nominal	**10**	Nominal	Not Null
15	Nominal	Nominal	Nominal	Nominal	**Positive Infinity**	Nominal	Not Null
16	Nominal	Nominal	Nominal	Nominal	Nominal	**Negative Infinity**	Not Null
17	Nominal	Nominal	Nominal	Nominal	Nominal	**100**	Not Null
18	Nominal	Nominal	Nominal	Nominal	Nominal	**Positive Infinity**	Not Null

Boundary Value Testing (Version 3)

Table 4: Variables and Values Tested (Version 3)

Variables	Minimum	Maximum	Nominal
Barrel Pose (x, y, z)	Negative Infinity	Positive Infinity	0,0,0
Radius	0	1	0 > radius <= 0.25
Mass	0	51	0 > mass <= 50
Initial Speed	0	Positive Infinity	10
External Force (x, y, z)	Negative Infinity	Positive Infinity	100, 100, 0

Table 5: Variables and Values Tested (Version 3)

Case	Barrel Pose (x, y, z)	Radius	Mass	Initial Speed	External Force (x, y, z)	Expected and tested Results (Position – x, y, z)
1	Negative Infinity	Nominal	Nominal	Nominal	Nominal	Not Null
2	0,0,0	Nominal	Nominal	Nominal	Nominal	98.52,194.76,0
3	Positive Infinity	Nominal	Nominal	Nominal	Nominal	Not Null
4	Nominal	0	Nominal	Nominal	Nominal	Throws Illegal Argument Exception
5	Nominal	0.25	Nominal	Nominal	Nominal	98.52,194.76,0
6	Nominal	1	Nominal	Nominal	Nominal	Throws Illegal Argument Exception
7	Nominal	Nominal	0	Nominal	Nominal	Throws Illegal Argument exception
8	Nominal	Nominal	10	Nominal	Nominal	98.52,194.76,0
9	Nominal	Nominal	51	Nominal	Nominal	Throws Illegal Argument Exception
13	Nominal	Nominal	Nominal	0	Nominal	Null
14	Nominal	Nominal	Nominal	10	Nominal	98.52,194.76,0
15	Nominal	Nominal	Nominal	Positive Infinity	Nominal	Not Null
16	Nominal	Nominal	Nominal	Nominal	Negative Infinity	Not Null
17	Nominal	Nominal	Nominal	Nominal	100, 100, 0	98.52,194.76,0
18	Nominal	Nominal	Nominal	Nominal	Positive Infinity	Not Null

Equivalence Class Testing

Equivalence Class Testing
Group members: Jasmeet Singh, Ahras Ali, Bulbul Arora

Table 1: Equivalence Classes

Test Functions	Variables being tested	Equivalence classes
testRadiusValue()	round_shot.radius	(-INF, 0) , [0] , [0.26, INF)
testMassValue()	round_shot.mass	(-INF, 0) , [0] , [50, INF)
testIntialSpeedNegativeValue ()	cannon.intialSpeed	(-INF, 0)
testExternalForceNegativeVlaue_XComponent()	cannon.externalForce[0]	(-INF, 0)
testExternalForceNegativeVlaue_YComponent()	cannon.externalForce[1]	(-INF, 0)
testExternalForceNegativeVlaue_ZComponent()	cannon.externalForce[2]	(-INF, 0)

- Equivalence Class Testing, also known as Equivalence Partitioning, is a black box software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.

- This technique reduces the total number of test cases that need to be developed, while still reasonably ensuring that all scenarios are covered.

Table 2: Equivalence Class Tests

Test method	Test Purpose	Expected Outcome	Actual Outcome	Test Pass
validInputforSimulator()	To check if the new simulator is created	The array of the position should not be null	The array of the position is not be null	Yes
testRadiusValue()	Test Negative Value for the Radius	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testRadiusValue()	Test zero Value for the Radius	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testRadiusValue()	Test Exceeding Value for the Radius	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testMassValue()	Test Negative Value for the Mass	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testMassValue()	Test Zero Value for the Mass	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testMassValue()	Test Exceeding Value for the Mass	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testIntialSpeedNegative Value ()	Test Negative value for the initial speed	IllegalArgument Exception should thrown	IllegalArgument Exception thrown	Yes
testExternalForceNegativeVlaue _XComponent()	Test negative x component of the external force	The position of the array should not be null	The array of the position is not be null	Yes
testExternalForceNegativeVlaue _YComponent()	Test negative y component of the external force	Test passes if no exception is thrown	The test passed	Yes
testExternalForceNegativeVlaue _zComponent()	Test negative x component of the external force	The position of the array should not be null	The array of the position is not be null	Yes

Decision Table Testing

Decision Table Based Testing
Group members: Ahras Ali, Bulbul Arora, Jasmeet Singh

Conditions:
1. 0.25 < Round_shot.radius > 0
2. 10 < Round_shot.mass > 0

Table 1: Decision Table for Round Shot Class

Conditions	0.25 < Round_shot.radius > 0	F	F	T	T
	10 < Round_shot.mass > 0	F	T	F	T
Actions	Check Mass	X		X	
	Check Radius	X	X		
	Valid Result				X

Conditions:
1. cannon.initialSpeed > 0
2. cannon.externalForce != [0,0,0]
3. 0.25 < Round_shot.radius > 0
4. 10 < Round_shot.mass > 0
5. inProgressSimulation == true

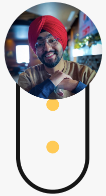
Table 2: Decision Table for Integration Testing

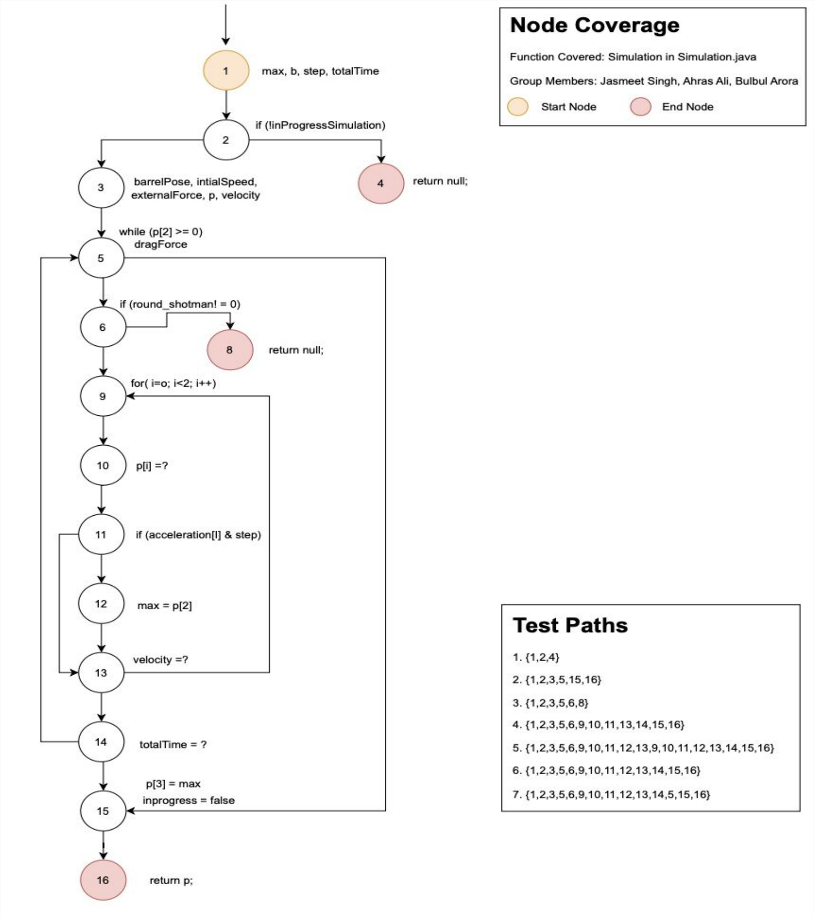
Conditions	cannon.initialSpeed > 0	-	F	T	T	T	T	F	T	T
	cannon.externalForce != [0,0,0]	-	T	F	T	T	T	F	T	T
	Round_shot.radius > 0	-	T	T	F	T	T	F	T	T
	Round_shot.radius < 0.25	-	T	T	T	T	T	T	F	T
	Round_shot.mass > 0	-	T	T	T	F	T	F	T	T
	Round_shot.mass < 10	-	T	T	T	T	T	T	T	F
	inProgressSimulation == true	F	T	T	T	T	T	F	T	T
Actions	Impossible	X						X		
	Check Radius				X				X	
	Check Mass					X				X
	Check Initial Speed		X							
	Check External Force			X						
	Output Position						X			

Decision Table Testing is a black box test design technique that uses a table to represent different combinations of inputs and their outputs.

In our project, we used black box testing to determine our conditions for the round shot class and the integration testing.

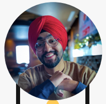
Node coverage

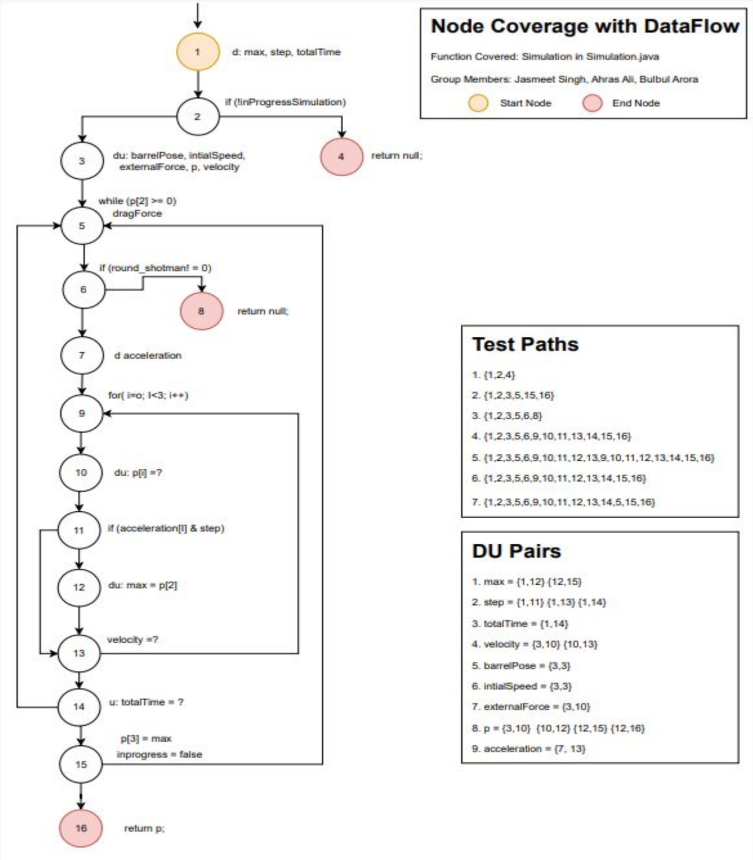




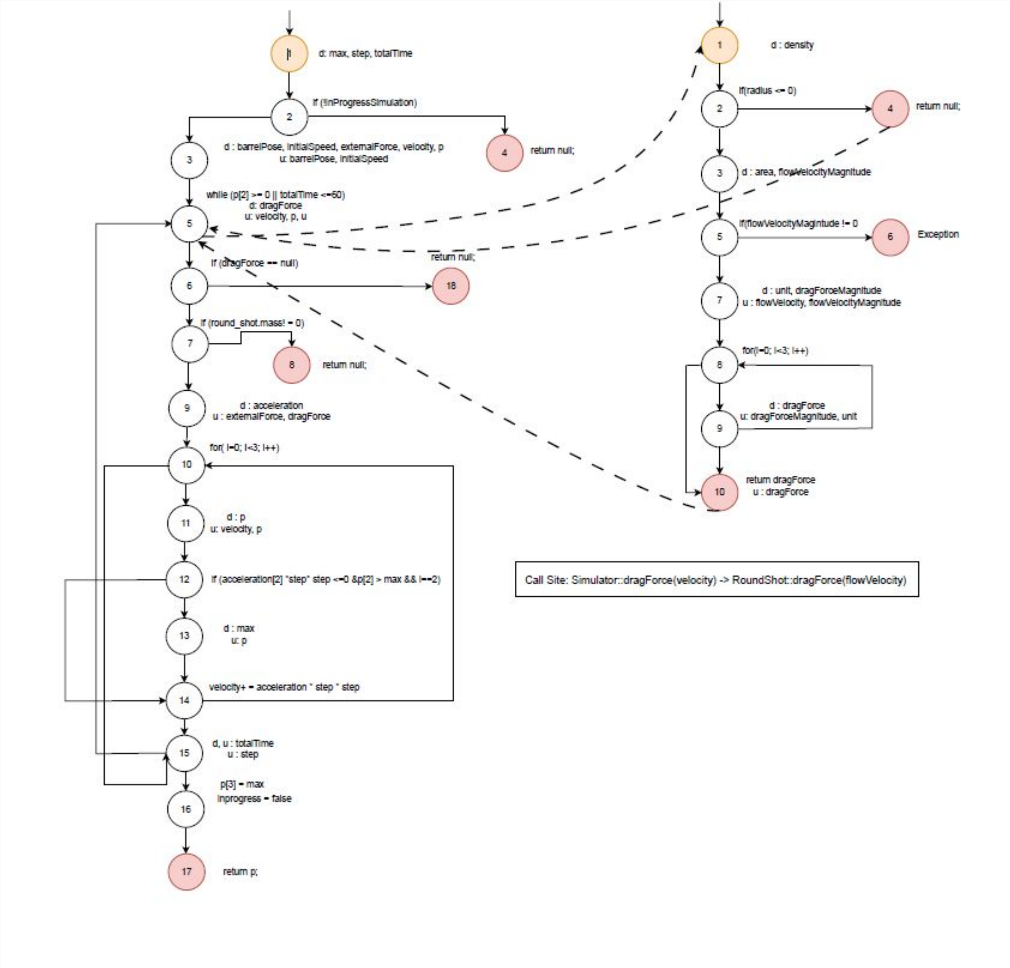
Node coverage testing in the context of data flow refers to a white-box testing technique that not only ensures each executable statement or node in the source code has been executed at least once, but also tracks the specific variables' values and the paths these variables take through the program.

Node coverage & Data flow





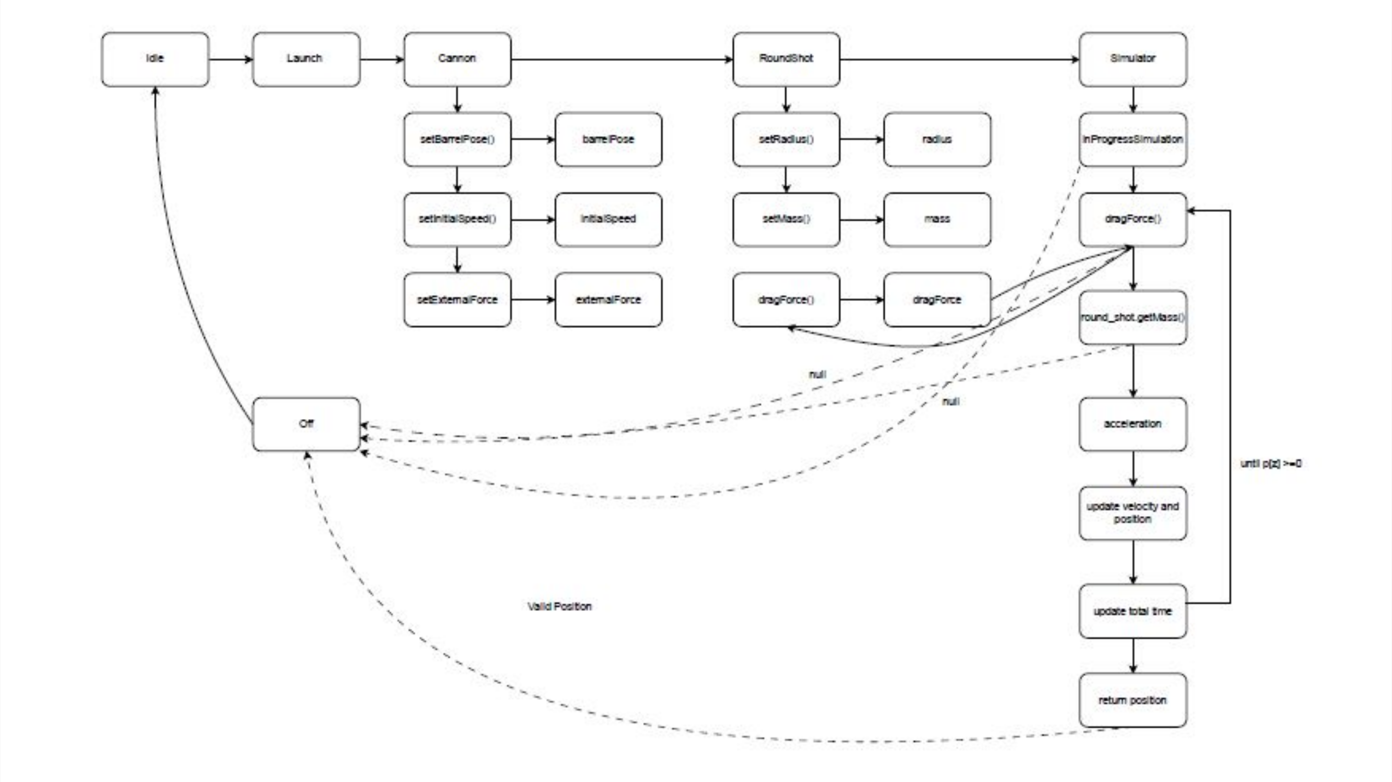
Data flow testing focuses on the points at which variables receive values (definition points) and the points at which these values are used (use points), along with the routes data takes between definition and use (DU paths).



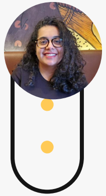
Integration Testing

We created a control flow graph for our Simulator::simulation() method and the RoundShot::dragForce() method.

FSM



We created an FSM to show what states our system can be in and how the state variables change the state of the system.



Class Diagram MVP 1

Simulator.java

- Gravity: Double
- barrelPose: double []
- radius : double
- mass: double
- dragCoefficient: double
- initialSpeed: double
- externalForce: double []

+ setBarrelPose (x:double, y:double, z:double): void
+ setRadius(radius: double): void
+ setMass(mass: double): void
+ setDragCoefficient(drag: double): void
+ setExternalForce (x:double, y:double, z:double): void
+ dragforce(flowVelocity: double []): double []
+ simulation(): double []

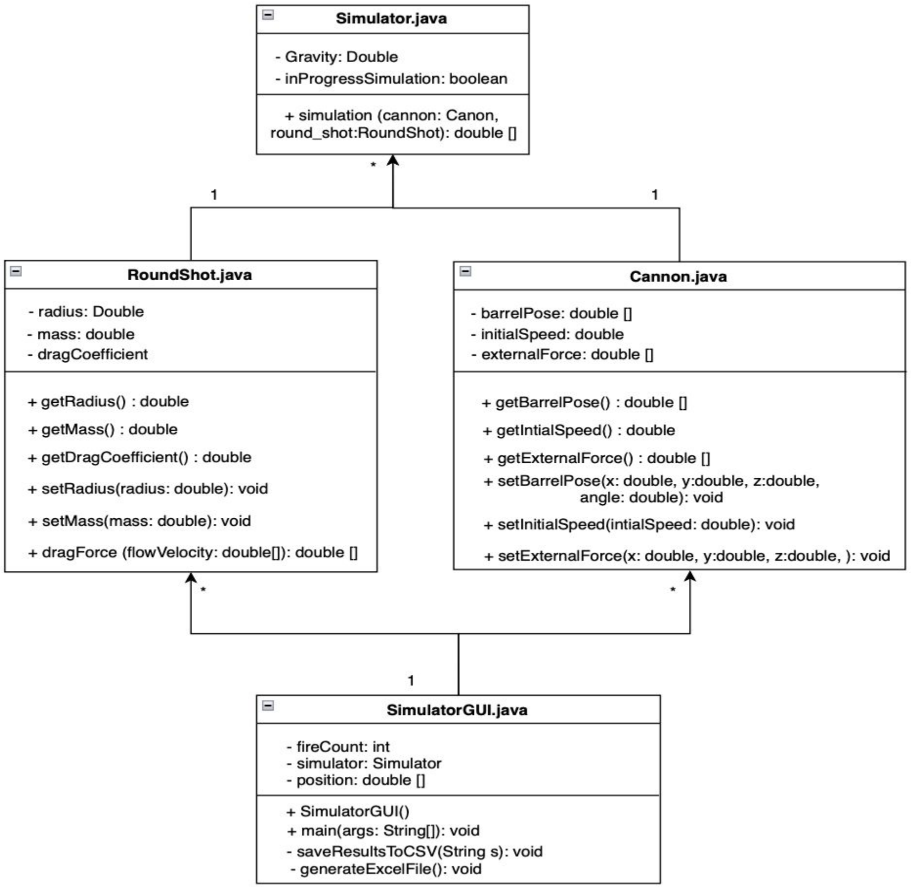
Class Diagram MVP 2

Simulator.java

- Gravity: Double
- barrelPose: double []
- radius : double
- mass: double
- dragCoefficient: double
- initialSpeed: double
- externalForce: double []

+ setBarrelPose (x:double, y:double, z:double): void
+ setRadius(radius: double): void
+ setMass(mass: double): void
+ setDragCoefficient(drag: double): void
+ setExternalForce (x:double, y:double, z:double): void
+ dragforce(flowVelocity: double []): double []
+ simulation(): double []

+ isValidRadius(radius: double): boolean
+ isValidMass(mass: double): boolean
+ isValidDragCoefficient(dragCoeffcient: double): boolean



Class Diagram
MVP 3

Weighted Matrix

Weighted Matrix:

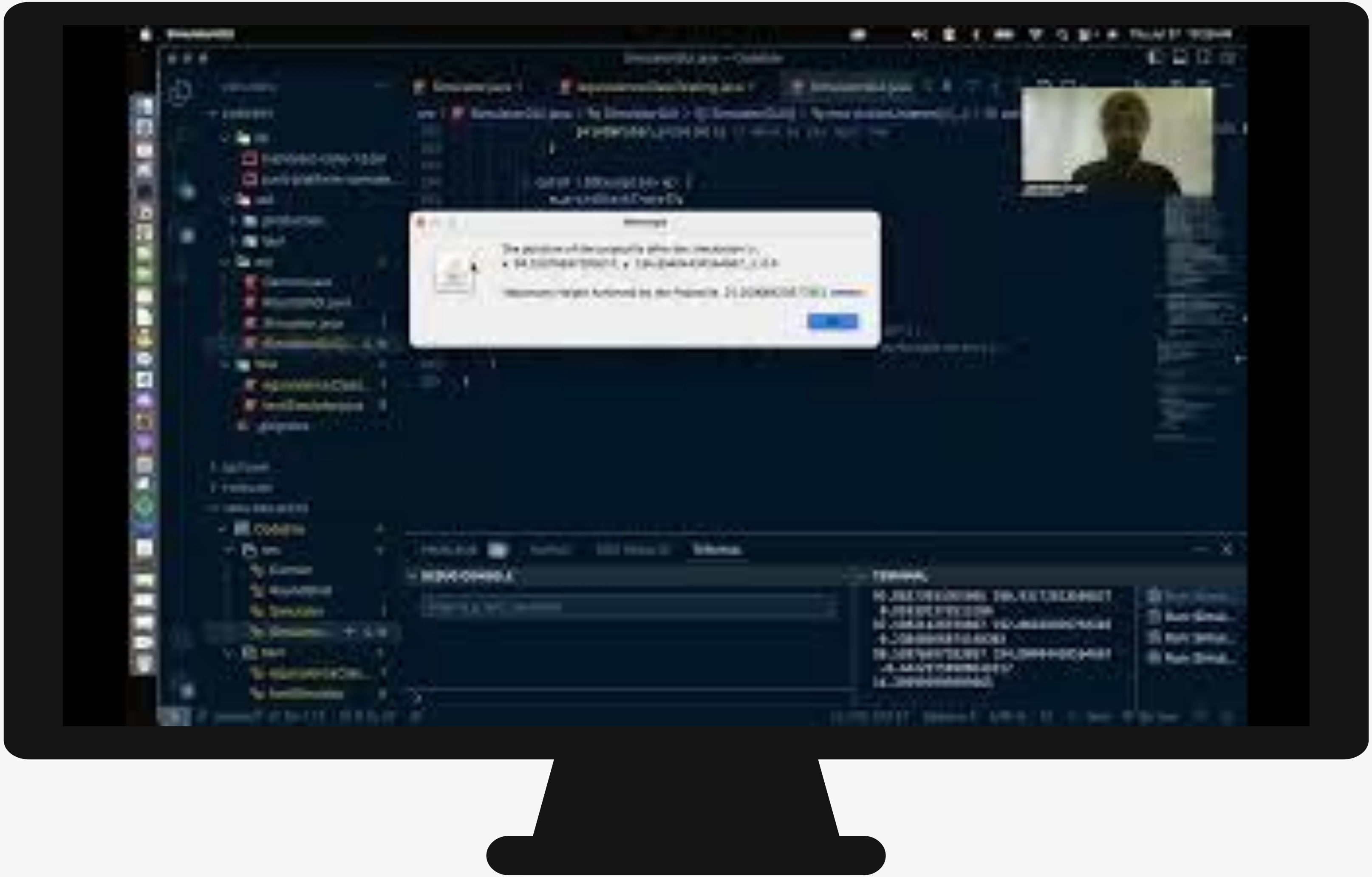
Criteria	Rating
Security	4
Accuracy	5

Version	Security (4)	Accuracy (5)	Total Score
Version 1	1 * 4	2 * 5	14
Version 2	5 * 4	3 * 5	35
Version 3	5 * 4	4 * 5	40

05

Results

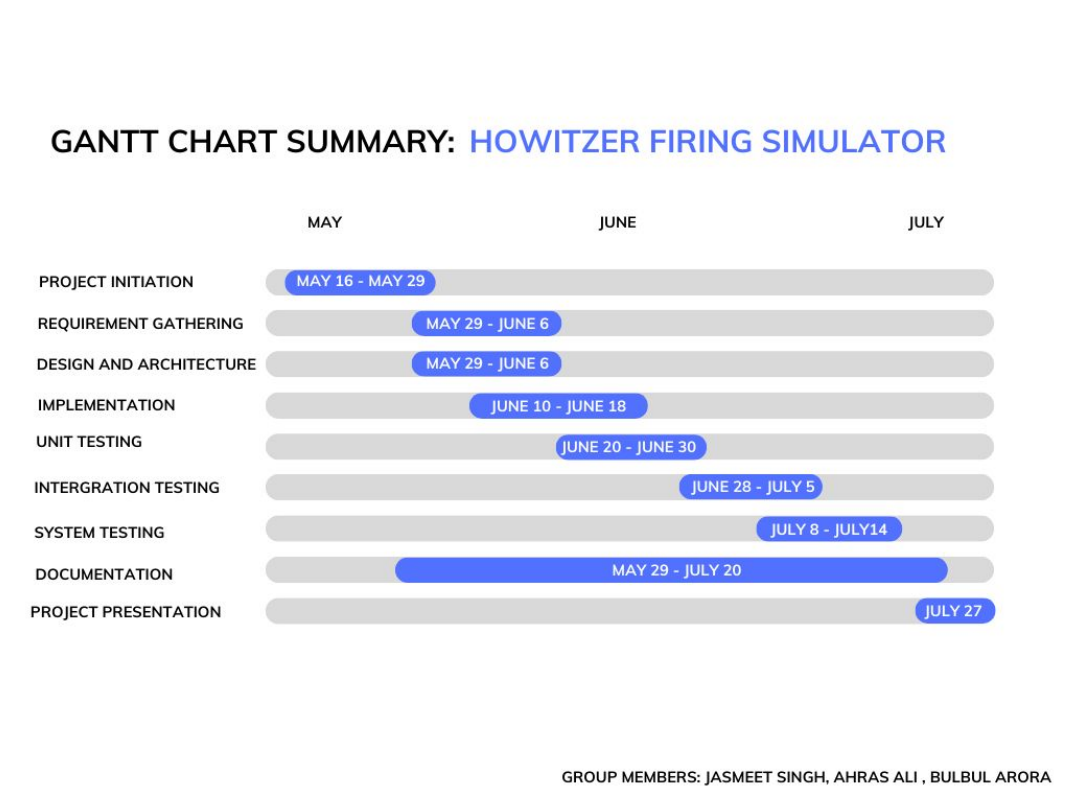
Demo



06

Project Management

Project management



07

Future Work

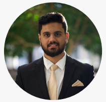
Future work

Model Specificity: Increase the accuracy of simulations by tailoring them to specific Howitzer models based on real-world data from the military.

Weather Considerations: Incorporate wind and rain factors to make the simulations more realistic and account for their impacts on Howitzer operation.

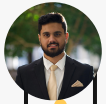
Augmented and Virtual Reality: Integrate AR and VR technologies to provide immersive training experiences, enabling users to interact with the simulator in a more engaging and realistic environment.

User Experience: Continuously refine and improve the user interface and experience, based on user feedback and testing, to make the simulator more intuitive and user-friendly.



08

Conclusion



Conclusions

Overview: Our team successfully developed a Howitzer simulator, focusing on both usability and testing.

Development: Throughout the development process, we used testing strategies to ensuring our application's reliability, robustness, and safety.

Approach: We approached the project in three MVP stages, starting MVP 1 from basic functionality to MVP 2 of safety measures, and finally, refining and expanding features for MVP 3.

Future: To further elevate the simulation experience, we plan to continuously improve our application for future use.

Closing Remarks: Our Howitzer simulator is more than a training tool; it's an easy to use, realistic, and safe environment for learning the Howitzers operations.

Thanks!

Do you have any questions?