# TRAIN YOUR FOES

-AVANISH BHARAT SALUNKE

B. TECH IN INFORMATION TECHNOLOGY

VJTI 2025 – 26

# ABSTRACT :

Most games have enemies that behave the same way every time as they follow fixed rules and don't really learn from how the player acts. In this project called "Train Your Foes," the goal is to create a game where the enemy AI observes the human player's behaviour, learns from it, and then becomes smarter with each round. The more predictable the player becomes, the better the enemy gets at countering them.

To achieve this, the project uses two main machine learning methods: Behavioural Cloning, where the AI tries to copy the player's actions based on past gameplay, and Reinforcement Learning, where the AI improves by trying different strategies and getting rewarded for better outcomes. Together, these techniques help the enemy first understand the player and then adapt to challenge them more effectively.

The game can be of any style such as a platformer, a top-down battle arena, or even a turn-based game. The focus is on building a system where the enemy changes its behaviour over time based on what it learns, instead of acting with fixed responses. This makes the gameplay more dynamic and interesting and shows how machine learning can be used in real-time game environments.

# ACKNOWLEDGEMENT :

I would like to thank ISHAAN SHAIKH brother and ABHAY VARNEKAR brother and all our seniors and mentors, for guiding me throughout this project. They were always helpful and gave useful feedback that made it easier for me to understand how to plan and write the proposal. Their experience really helped me improve my ideas and approach.

I'm also thankful to Project X for giving us this opportunity. Working on this project helped me explore something new and exciting, and it encouraged me to learn more about how AI can be used in games.

Lastly, I want to thank my friends and peers for their support and motivation during this process. Even though this is just the proposal stage, I've already learned a lot and enjoyed working on it.

# CONTENTS :

Section 1

# PREFACE :

## ABOUT ME

My name is Avanish Bharat Salunke. I am currently in my first year at VJTI, and soon going into my second year as holidays are going on. I have a strong interest in game development and I really enjoy learning how games work and how to make them.

I have a good hold on C++ and Python, and I use them for most of my coding. I'm a beginner in machine learning, but I really want to learn it because it feels exciting and useful, especially in games. I've also done some basic work in OpenCV and I enjoy it, I'm not an expert, but I find it interesting and fun to use.

This project felt like the right choice for me because it mixes AI and games, both of which I want to explore more. I hope to learn a lot through this and improve my understanding of how smart game systems can be built.

## MOTIVATION FOR THIS PROJECT :

Most games today have enemies that behave in a fixed and predictable way. Once the player figures out their patterns, it becomes easy to win, and the challenge slowly fades away. This project, Train Your Foes, is interesting because it focuses on creating an enemy that doesn't just follow a script it learns from the player's actions and adapts over time.

The idea of an AI that can observe how a player plays, find patterns in their behaviour, and then use that to improve itself is something that can completely change how games feel. It can make the experience more dynamic, challenging, and personalized. Instead of fighting the same enemy logic over and over, the player would be forced to change their strategy constantly, which makes the gameplay a lot more engaging.

Using techniques like Behavioural Cloning and Reinforcement Learning, the AI in this project will first copy the player's behaviour and then gradually learn how to counter it. This makes the enemy feel more human-like and unpredictable. It also opens interesting possibilities for building smarter game systems that react to each individual player differently.

What makes this project more exciting is that it's open-ended in terms of gameplay. The core focus is on the learning part of the AI, so the game can be of any type- a platformer, a top-down shooter, or even a turn-based setup. This gives a lot of room for creativity while working on a meaningful problem that combines game development with machine learning.

## OBJECTIVE OF THIS PROJECT :

The main objective of this project is to build a game where the enemy AI does not follow a fixed logic, but instead learns from the player's behaviour over time and adapts its strategy to become more challenging. The idea is to move away from static enemy actions and make the gameplay more dynamic, where the AI gets better the more you play against it.

To achieve this, the project combines concepts from machine learning, especially behavioural cloning, and reinforcement learning, and connects them with a real-time game environment. The enemy will first observe how the player behaves, then try to imitate it, and finally learn how to counter the player using a reward-based learning system. The game becomes more difficult not because of hardcoded difficulty levels, but because the AI itself improves with experience.

## PROJECT FLOW :

The project flow outlines the step-by-step process used to build an adaptive enemy AI that learns from the player’s behaviour. Starting with player action logging and moving through behavioural cloning, reinforcement learning, and real-time integration, each stage plays a key role in making the enemy smarter and more challenging over time. The flow ensures a smooth connection between gameplay and AI learning.

Section 2

# THEORY AND APPROACH :

## A. DESIGNING THE GAME ENVIRONMENT

**What is a game engine ?**

A game engine is a software framework used to build and run video games. It provides all the essential tools and systems needed to create interactive experiences, including rendering graphics, handling user input, managing physics, playing audio, and organizing game logic. By using a game engine, developers can focus more on designing gameplay rather than building low-level systems from scratch. For this project, the game engine acts as the foundation where both the human player and the AI enemy interact in real time.

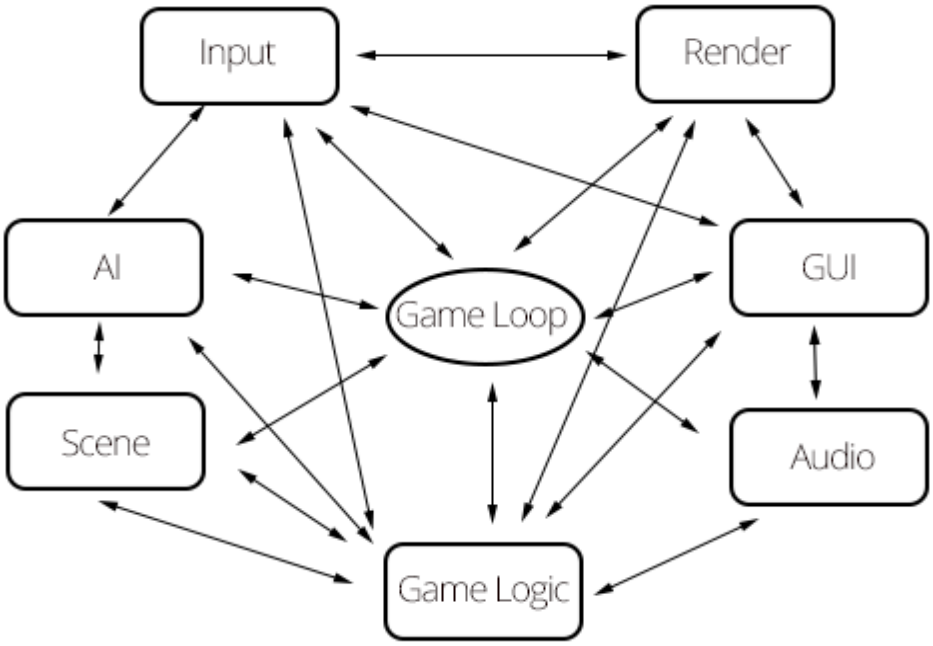
**The basic functionalities of a game engine are :**

- A rendering engine, for 2D or 3D graphics.
- Input handling (for keyboard & mouse, touch devices etc).
- Game loop (the internal routine than recalculates game events every frame).
- A physics engine, with collision detection and response, sound.
- Animations (for 2D sprites or 3D models).
- Memory management.
- Process threading (allowing for multiple, parallel processes).

**The additional functionalities are :**

- Scripting;
- Artificial intelligence;
- Networking;

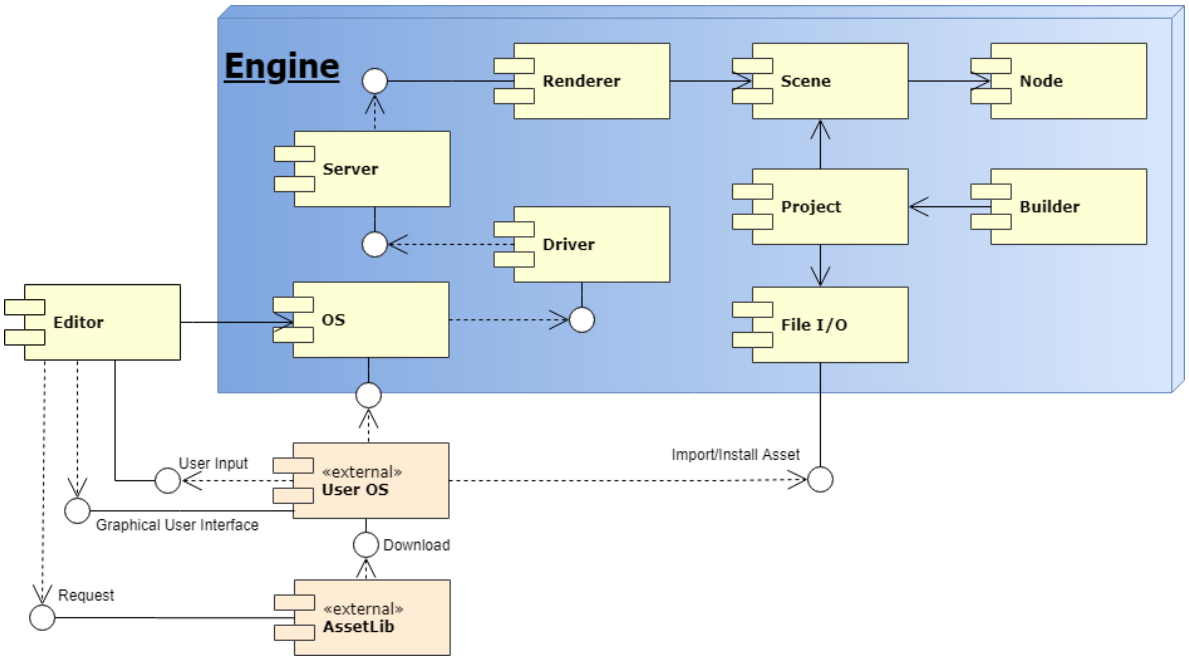
- Streaming;
- Localization support;
- Multi-platform publishing.




**2.1 : Basic working of a game engine**

**GODOT / GDSCRIPT**

Godot is an open-source, cross-platform game engine known for its lightweight design, flexible node-based architecture, and built-in support for 2D and 3D game development. It uses GDScript, a Python-like scripting language, which makes it beginner-friendly and easy to extend. Godot's modular scene system, built-in physics, animation tools, and input handling make it an ideal choice for creating custom game environments while keeping full control over gameplay logic. Its open nature also allows easy integration with external systems like Python-based AI models, making it suitable for projects involving machine learning or adaptive game mechanics.




**2.2 : FUNCTIONAL STRUCTURE OF GODOT ENGINE**

**Why am I thinking GODOT over any other game engine????**

1. **Free ‘er’ than the college WIFI.**

   Unity: "Pay up."

   Unreal: "Your soul, please."

   Godot: "Bro I'm free, open-source, and don’t even want your email. Just make out(the project) and chill"

2. **Modular NODE system : just like ~~PROJECT X~~ SRA projects (broken but adjustable)**

   Godot’s flexible, OOP-based design using Nodes and Scenes, ideal for modular game structure, asset reuse, and clarity in scripting.

3. **Its Script is like a CP lab language we deserved but never got it. GDSCRIPT is developed on python.**

4. **AI connection smoother than ~~PROJECT X~~ VJTI website.**

5. **Has a specific 2D focus because we just care about passing and placement.**

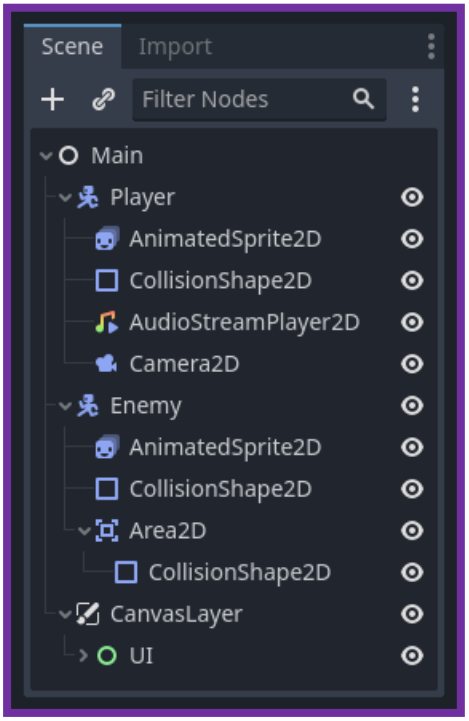
**Some of the main features of GODOT that I am going to use :**

1) **NODE – SCENE system :**

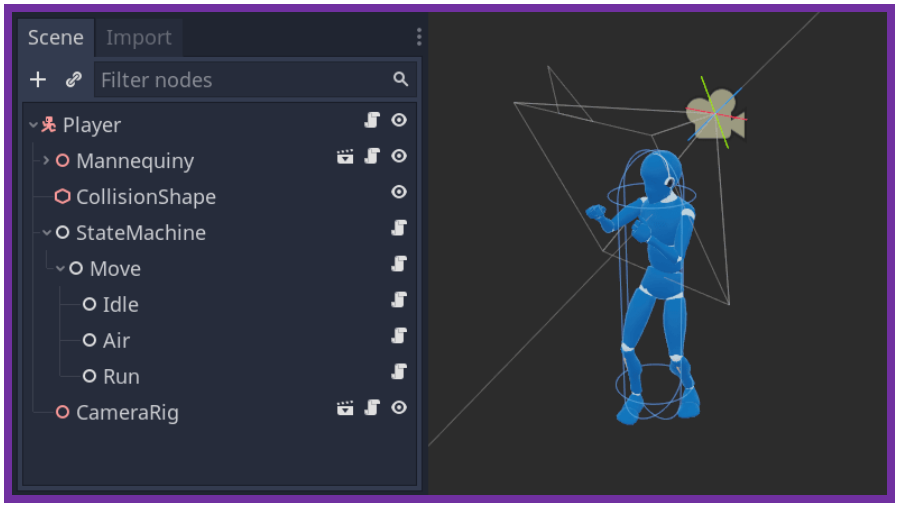
NODES are the fundamental building blocks of your game. They are like the ingredients in a recipe. There are dozens of kinds that can display an image, play a sound, represent a camera, and much more. You can add a node to another node as a child.

When you organize nodes in a tree, like our character, we call this construct a SCENE. Once saved, scenes work like new node types in the editor, where you can add them as a child of an existing node. In that case, the instance of the scene appears as a single node with its internals hidden.

Scenes allow you to structure your game's code however you want. You can compose nodes to create custom and complex node types, like a game character that runs and jumps, a life bar, a chest with which you can interact, and more.




**2.3 : BASIC NODES IN A SCENE**

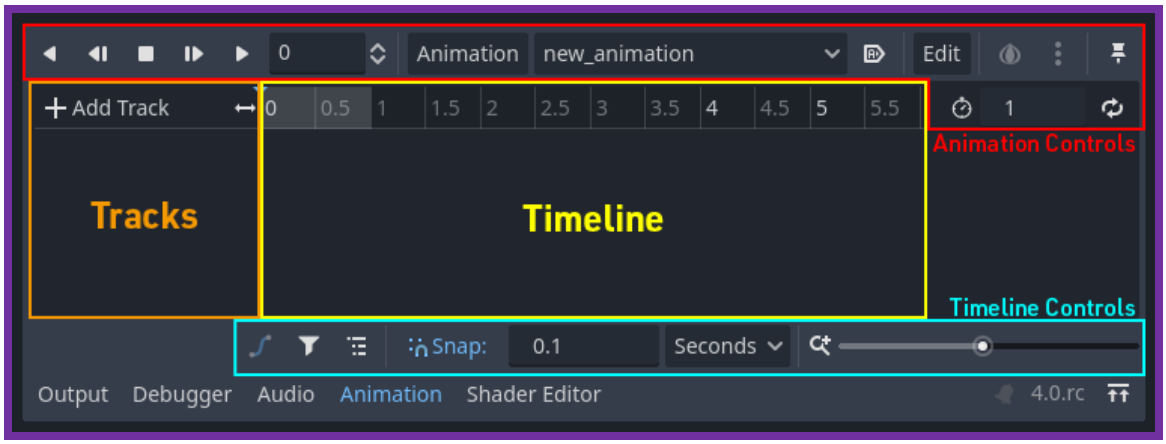



**2.4 : GODOT SCENE EDITOR**

## 2) ANIMATIONS IN GODOT :

The AnimationPlayer node allows you to create anything from simple to complex animations.

The AnimationPlayer node type is the data container for your animations. One AnimationPlayer node can hold multiple animations, which can automatically transition to one another



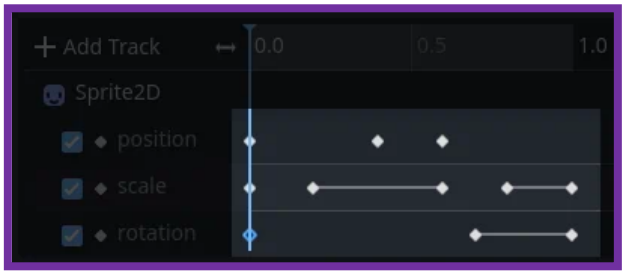

**2.5 : GODOT ANIMATION INTERFACE**

**The panel consist of 4 parts :**

- Animation controls(i.e. add, load, save and delete animations)
- The tracks listing.
- The timeline with keyframes.
- The timeline and track controls, where we can zoom the timeline and edit tracks.

**KEYFRAMES :**

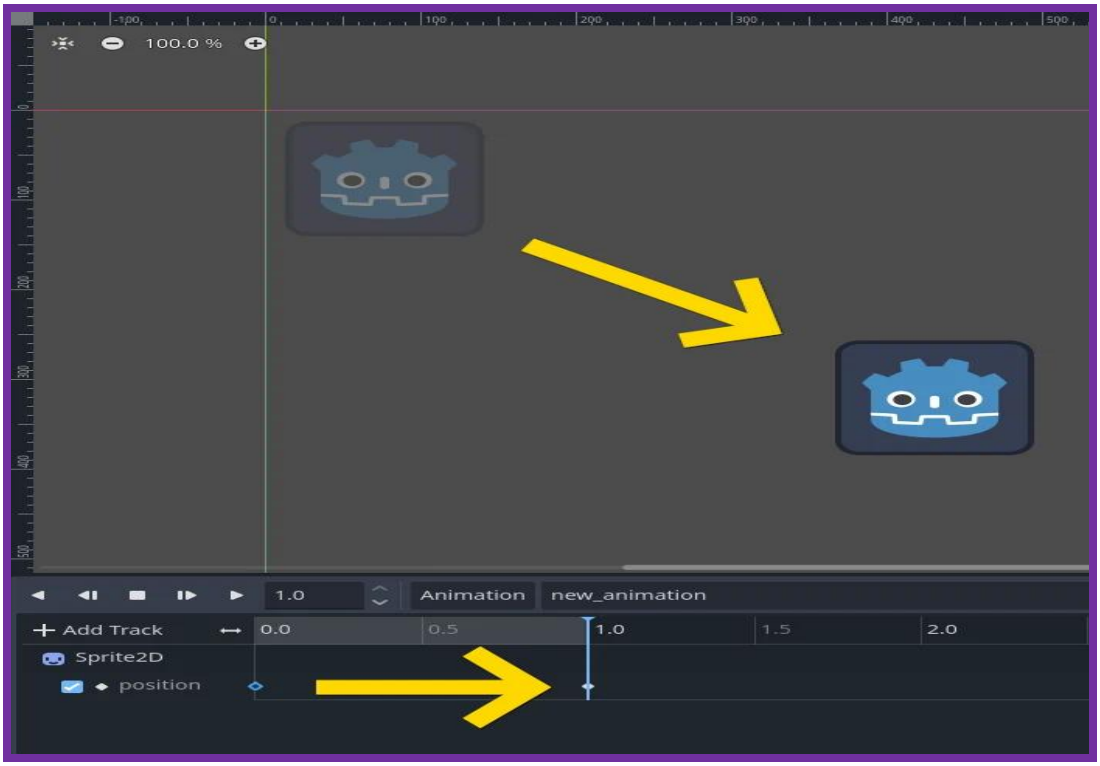
A keyframe defines the value of a property at a point in time. Diamond shapes represent keyframes in the timeline. A line between two keyframes indicates that the value doesn't change between them. Keyframes set up the required stage for animation




**2.6 : KEYFRAMES**

To animate using keyframes in Godot, first add an AnimationPlayer node to your scene. Then, create a new animation and select the object you want to animate. Move to a specific frame on the timeline, adjust the property you want to animate (like position, rotation, or scale), and click the key icon to insert a keyframe. Repeat this at different frames with different values to create

movement or transitions. When the animation is played, Godot smoothly interpolates between the keyframes, creating the final animated effect.




**2.7 : FINAL ANIMATION**

### 3) GDSCRIPT :

GDScript is a high-level, object-oriented, imperative, and gradually typed programming language built for Godot. It uses an indentation-based syntax similar to languages like Python. Its goal is to be optimized for and tightly integrated with Godot Engine, allowing great flexibility for content creation and integration.

GDScript is completely independent from PYTHON and its not based on it. But, it is inspired by PYTHON through its syntaxes and styles.

- GDScript uses indentation just like PYTHON.
- It has a clean, readable syntax.
- It uses dynamic typing means we can declare variables without defining the type.
- Its supports classes, functions, if-else loops that are all similar to PYTHON.

```
class_name MyClass

# Inheritance:
extends BaseClass


# Member variables.
var a = 5
var s = "Hello"
var arr = [1, 2, 3]
var dict = {"key": "value", 2: 3}
var other_dict = {key = "value", other_key = 2}
var typed_var: int
var inferred_type := "String"

# Constants.
const ANSWER = 42
const THE_NAME = "Charly"

# Enums.
enum {UNIT_NEUTRAL, UNIT_ENEMY, UNIT_ALLY}
enum Named {THING_1, THING_2, ANOTHER_THING = -1}

# Built-in vector types.
var v2 = Vector2(1, 2)
var v3 = Vector3(1, 2, 3)


# Functions.
func some_function(param1, param2, param3):
    const local_const = 5

    if param1 < local_const:
        print(param1)
    elif param2 > 5:
        print(param2)
    else:
        print("Fail!")

    for i in range(20):
        print(i)
```

**2.8 : GDScript**

Paint Protocol is a fast-paced 3D first-person paintball shooter game where strategy meets reflex. Designed to simulate a tactical paintball arena, players engage in vibrant solo combat with realistic movement and weapon dynamics.

**GAMEPLAY MECHANICS :**

- Movement : Free navigation in a 3D arena with support for running, crouching, jumping and dodging obstacles.
- Combat : Color-based paintball guns with splash effects and hit decision.
- Resource Management :
    I. Refill paint ammo.
    II. Switch paint guns with varying speeds, splash sizes an accuracy.
- Defence : Crouch behind cover, strafe, or retreat to safer zones.

- Health System : Health bar reduces on hits, regenerates slowly or via pickup.
- Scoring system : Based on accuracy, survival time, and objective completion.

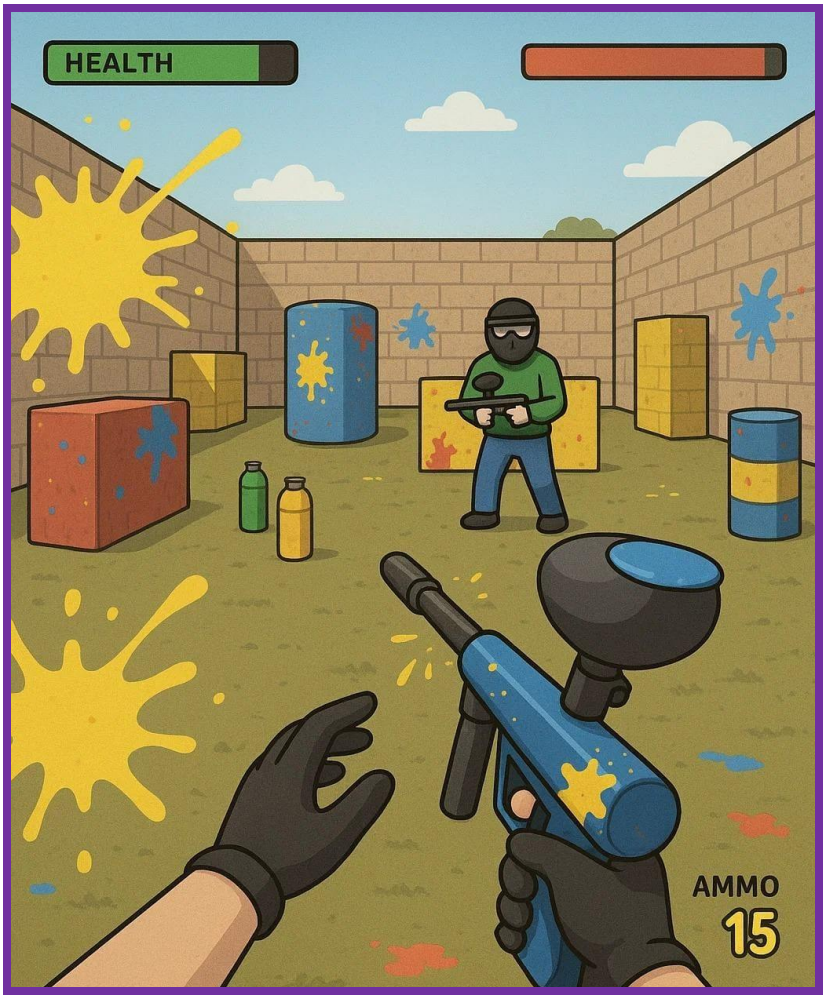
Paint Protocol is not just a game it's a testbed for intelligent enemy behaviour. Every match trains your foe to fight smarter. Whether you're sneaking behind barriers or charging in guns blazing, the enemy is always one step closer to figuring you out.

**AI INTEGRATION :**

What makes Paint Protocol unique is the presence of a learning-based AI enemy:

- It doesn't follow static patterns.
- It observes how the player behaves.
- It adapts using Behavioral Cloning, Pattern Detection, and Reinforcement Learning.

As players fall into habits (like rushing while low on ammo or crouch-shooting near corners), the AI learns to exploit these predictable tactics making every rematch feel smarter and more challenging.




**2.9 : SAMPLE GAMEPLAY**

# B. IMPLEMENT PLAYER BEHAVIOUR LOGGING

**What exactly am I thinking to do???**

To train the AI what we need to do is firstly understand what our player is doing, for that scene will just store the information such as the player's information like position, velocity, other kinds of things like any defence system, its position and all in the form of JSON file, this process is called as logging.

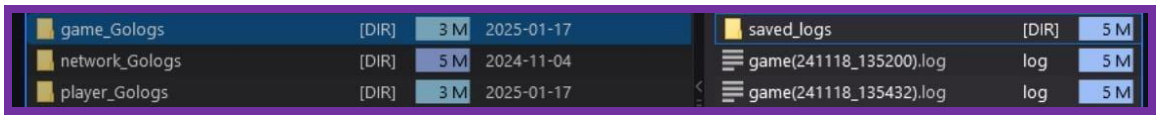

**JSON ( JavaScript Object Notation )**

JSON, also known as JavaScript Object Notation, is a file format. We use JSON to store and maintain data in a human-readable text format. The format consists of attributes and their data types stored in the form of an array. The following is an example of a JSON array.

```
"employee":  {
      "name": "Rob",
          "age": "35",
          "salary":  5600,
          "City": "New York",
          "married": true
          }
}
```

Currently, the use of artificial intelligence (AI) is increasing. AI can monitor our log files and find anomalies. But for log files to be machine-readable, they need to come in a structured format. And what could be better than making the logs structured in JSON format? JSON files are easily readable and for making the JSON files directly from our scripted game, we will be using GoLogger.

GoLogger is a simple yet flexible logging framework for Godot 4.3+, which captures and stores game events (and data) into external .log files accessible by both developers and players to aid in development and maintenance of your projects. With minimal setup, it can be quickly integrated into any project to always run in the background or log specific events during testing.




**2.10 : GAME DIRECTORY**

**Why using this instead of normal logging???**

"If something saves me 10 lines of code, prevents 3 bugs, and reduces one mental breakdown, it's staying in the project."

-Engineer

**What does GoLogger do and what we want to do exactly???**

**FOR EXAMPLE :**

```
func _on_player_jump():
    logger.log("player_jump", {
        "x": player.position.x,
        "y": player.position.y,
        "action": "jump"
    }, 1)
```

- This logs in the data only when the player jumps or so.
- But what we actually want is to log in the data frame by frame so that our enemy can get a clear and a huge actions database to study the behaviour of the player.

```
func _physics_process(delta):
    logger.log("frame_data", {
```

```
        "frame": Engine.get_frames_drawn(),
        "player_x": player.position.x,
        "player_y": player.position.y,
        "action": current_action
    }, 1)
```

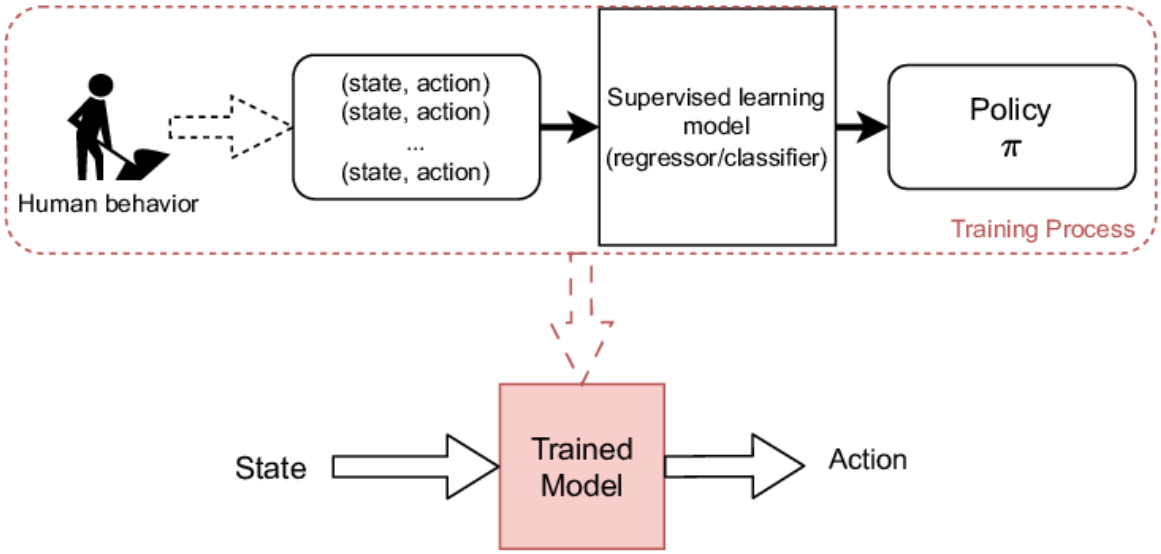
- We moved the logger.log() function inside _physics_process(), so it runs every frame, not just during specific actions or events.

**What will be the JSON file contain ???**

```
{
 "event": "frame_data",
 "timestamp": "19:17:27",
 "data": {
  "frame": 101,
  "player_x": 215,
  "player_y": 430,
  "action": "jump"
 }
}
{
 "event": "frame_data",
 "timestamp": "19:17:28",
 "data": {
  "frame": 102,
  "player_x": 220,
  "player_y": 428,
  "action": "fall"
 } }
```

# C. TRAIN A BEHAVIORAL CLONING MODEL

Now, we need to make a model that learns to mimic player behaviour using supervised learning. This model observe the game states and later helps us to predict the players actions, forming the base intelligence for the enemy.




**2.11 : PROCESS OF BEHAVIORAL CLONING**

**What is this policy π ???**

In the process of imitation learning, a policy is just a function that tells the AI enemy what action needs to be taken in response to the given state of the game.

$\pi(s) \rightarrow a$

(s) is the current state of the game.

(a) is the action that needs to be performed.

Consider a dataset $D = \{(s_i, a_i)\}^{M}_{i=1}$ consisting of state-action pairs collected from an expert policy $\pi^*$ , where $s_i$ represents the state and $a_i$ the action taken by the expert in that state. The objective of behavioral cloning is to learn a policy $\hat{\pi}$ that approximates the expert policy $\pi^*$ as closely as possible. The process involves:

1. Data Collection: Collect a dataset D of state-action pairs (s, a) by observing an expert performing the task.
2. Learning: Train a model $\hat{\pi}$ on D to learn the mapping from states to actions. This typically involves minimizing a loss function over the dataset

**POLICY BUILDUP IN PAINT PROTOCOL :**

In our game, the state(s) could include : player's position, ammo count, health level, weapon type, proximity to cover or enemy, whether the player is repeating a known pattern.

The action(a) might be : Move left/right, jump, crouch , shoot, switch gun, take cover etc.

We collect a dataset :

$D = \{(s_1, a_1), (s_2, a_2), ..., (s_M, a_M)\}$

Where:

- $s_i$ is a state from gameplay logs
- $a_i$ is the action the player took

Then we train a model $\hat{\pi}$ (pi-hat) to mimic the player's behavior:

- Input → state(s).
- Output → predicted action(a).

This is done by minimizing a loss function, usually cross-entropy (for classification) or MSE (for regression).

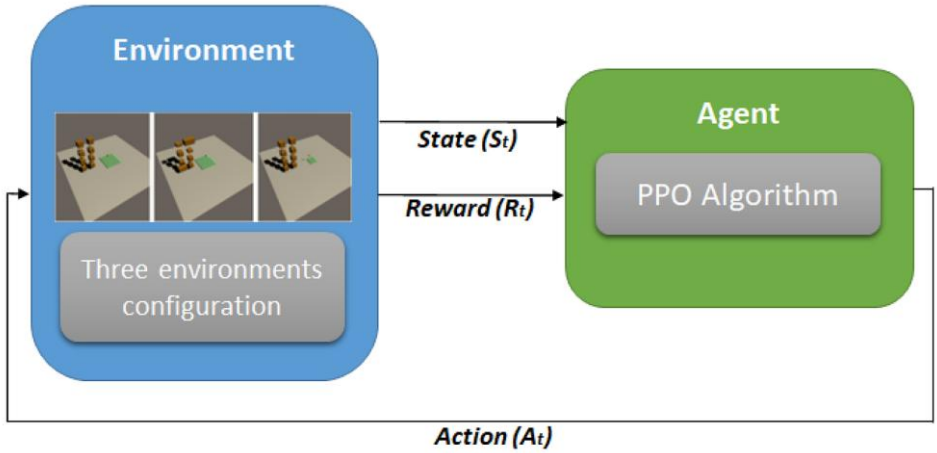
**After training:**

- $\hat{\pi}$ behaves like the player.
- When deployed in the game, the AI enemy acts like a clone of the player.
- This becomes the starting behaviour before R

**Why is there a need of policy optimization ???**

- If we don't optimize the policy, the AI will just crouch in a corner and stare at a wall the whole match. Basically, the game becomes VJTI depression simulator.

-




**2.12 : BASIC OPTIMIZATION WORKFLOW**

Proximal Policy Optimization (PPO) is a method that helps an agent improve its actions to get better rewards. Like other policy gradient methods, it directly changes how the agent makes decisions. PPO adds extra control to keep those changes from being too big.

It does this using a "clipping" rule which limits how much the policy can change at once. This helps keep learning safe and stable. The main goal of PPO is to find a balance between two things:

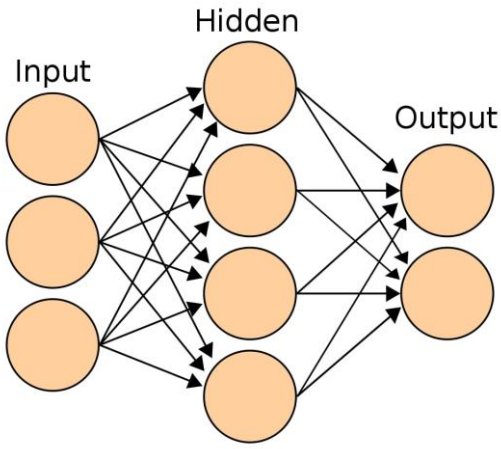
1. **Maximizing the objective:** This is the core of policy optimization where the agent's policy is adjusted to maximize expected rewards.
2. **Keeping updates small:** Big changes can mess up learning so PPO makes sure each update is limited to a safe amount to avoid problems.

The clipped objective helps ensure that the policy does not change too drastically between updates led to more stable learning and better overall performance.

**MLPClassifier : (multi – class classification)**

The MLPClassifier (Multilayer Perceptron Classifier) is a class from the Scikit-learn library that allows you to build and train a feedforward artificial neural network for classification tasks.

- An input layer to input the current game state.
- One or more hidden layers to learn patterns.
- An output layer to display the prediction of the enemy.




**2.13 : NEURAL NETWORK**

In deep learning, a MULTILAYER PERCEPTRON (MLP) is a name for a modern feedforward neural network consisting of fully connected neurons with nonlinear activation functions, organized in layers, notable for being able to distinguish data that is not linearly separable.

The MLP consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes. Since MLPs are fully connected, each node in one layer connects with a certain $w_{ij}$ to every node in the following layer.

**Why are we using multi-layer classification instead of normal binary classification ???**

- Binary is for 2 outputs i.e. yes/no or jump/notjump but the thing is multi-class is for 3 or more actions.
- Binary uses 1 output neuron, multi-class uses one per class.
- Binary gives one probability, multi-class gives probabilities for all actions.
- Our game needs multi-class since it has many possible actions.

**MLPClassifier can be used for beginning purposes but it might not be used for later purposes :**

A. **NO GPU SUPPORT**

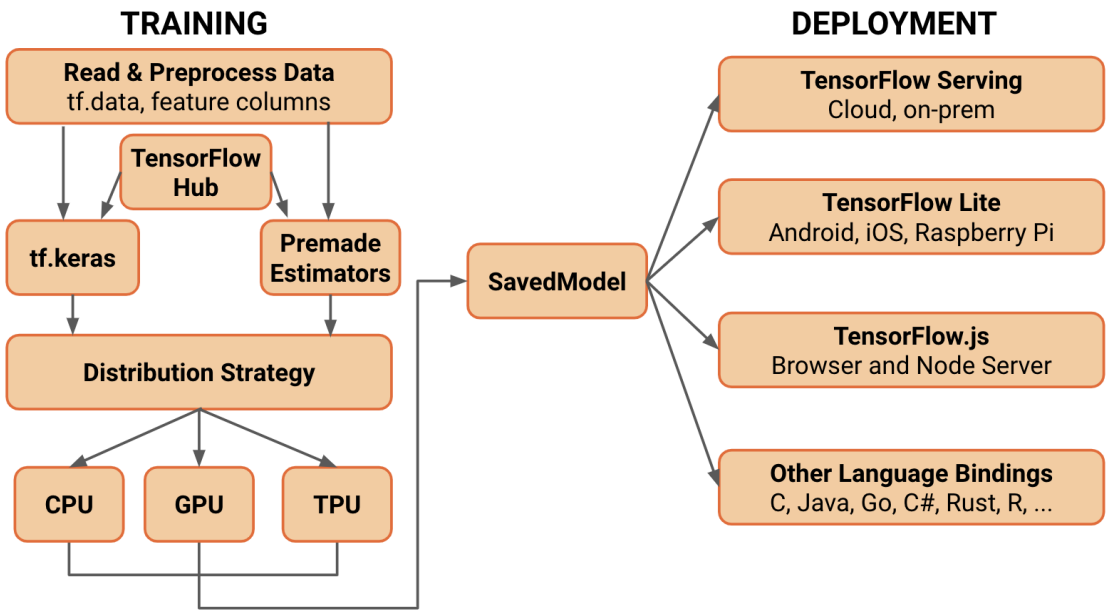
Using MLPClassifier on big data is like asking your college wifi to download GTA 6, too slow and painful

B. **NO REAL-TIME USE**

It may not run the final FLASK server properly.

C. **ZERO FLEXIBILITY**

Very few additions of dropouts, custom layers , change of loss functions




**2.14 : TENSORFLOW AND KERAS WORKFLOW**

**To overcome some of the limitations of the MLPClassifier, we might have to use TENSORFLOW instead of using it..**

**We define a Multilayer Perceptron (MLP) using tf.keras.Sequential, which allows us to:**

- Customize hidden layers (number, size, activation functions)
- Choose suitable loss and optimizer

**FOR EXAMPLE :**

The following snippet network learns to map game state vectors (inputs) to player actions (outputs), just like with MLPClassifier, but now with greater scalability :

```
model = keras.Sequential([
  keras.layers.Dense(64, activation='relu', input_shape=(state_dim,)),
  keras.layers.Dense(64, activation='relu'),
  keras.layers.Dense(num_classes, activation='softmax')
])
```

**INITIAL STEPS :**

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, batch_size=32)
```

**TRIANING THE MODEL :**

```
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']  )
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.1)
```

**SAVING THE MODEL :**

After training, the model is saved in .h5 format using model.save(), making it reusable for real-time inference. This saved model can be integrated into the game through a Flask API.(which wd be discussed at the very end.)

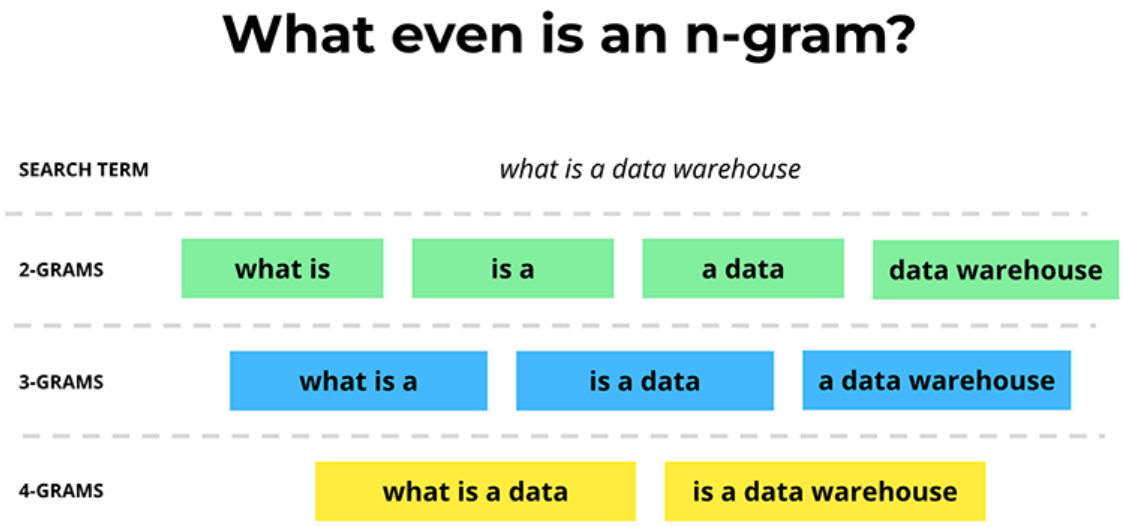
```
model.save("bc_model.h5")
```

# D. PATTERN DETECTION LOGIC

**This is the basic flow of this step :**

- Analysing JSON logs of the player actions collected in STEP B.
- Extracting repeated action patterns.
- Finding sequences that happen often.
- Feeding that info to the reinforcement learning stage so the AI can exploit predictable behaviour.

**N-GRAM ANALYSIS FOR PLAYER PATTERN DETECTION :**




**2.15 : N-GRAM IN A SENTENCE**

To detect repeating behaviour patterns in the player's actions, we apply a technique known as n-gram analysis. An n-gram is a continuous sequence of 'n' items, in our case, player actions extracted from gameplay logs. By breaking down a long sequence of actions into smaller overlapping chunks (n-grams), we can identify the most frequently occurring action sequences that represent the player's habits or tendencies.

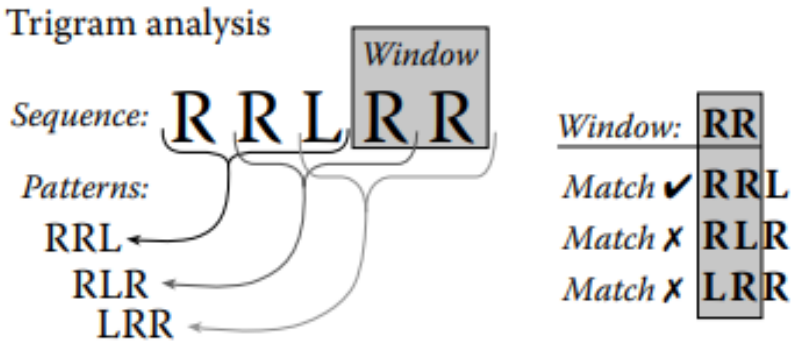
**FOR EXAMPLE :**

An N-gram predicts by picking out an observed pattern that it thinks is being executed again. To do so, it considers the most recent events in the sequence, and matches them against previously observed patterns. This set of recent events we'll call the window.

The length of the window is always N-1, so each pattern includes one more event than

the window length. Patterns match the window if their first N-1 events are the same. All matching patterns will have a unique Nth event. The pattern with the most occurrences has the highest probability, so its Nth event becomes the prediction. No division as such is required.

Let's see what our N-grams will predict next in the same sequence, "R, R, L, R, R."




**2.16 : EXAMPLE**

**Using N-gram statistics, we can make the following predictions:**

- ✓ The unigram predicts Right, since it uses raw probability. Its window size is N – 1 = 0.
- ✓ The bigram has observed two patterns that match its window, RR and RL. Since RR occurred once more than RL, the bigram chooses RR and predicts Right.
- ✓ The trigram finds that only RRL matches its window, so it uses RRL to predict Left.
- ✓ The 4-gram has not observed any patterns that start with LRR, so it cannot make a prediction.

**Why we specifically need n-gram analysis for this project ???**

This step applies n-gram-based pattern detection to identify frequently repeated sequences of player actions from gameplay logs. These patterns are analysed to detect behavioral habits and are later used to inform and guide the enemy AI's decision-making through reinforcement learning. The goal is to build an adaptive enemy that doesn't just react but predicts.

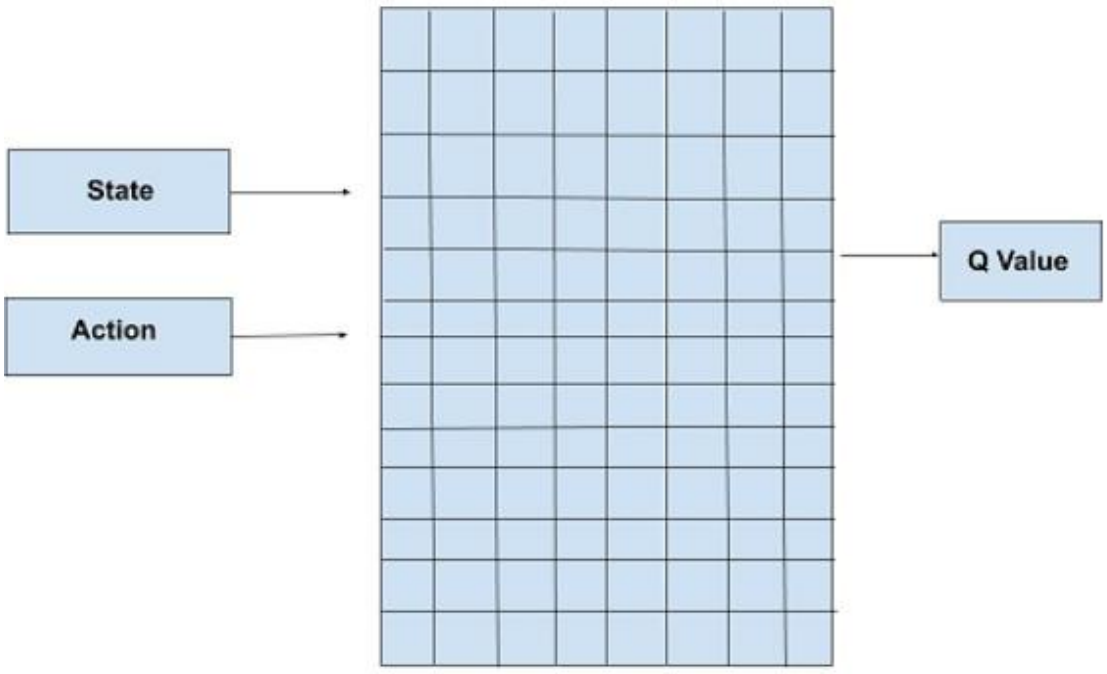
While pattern detection reveals frequent player behaviours, the reinforcement learning model is not explicitly told how to counter them. Instead, the detected patterns serve as additional input features or reward triggers. The AI learns its own counter-strategies through repeated interaction with the game environment, making the system adaptive rather than rule-based.

# E. REINFORCEMENT LEARNING (RL) MODEL

We train the AI enemy using Reinforcement Learning algorithms such as Deep Q-Network (DQN). The AI plays repeated matches against the player or a simulation and receives rewards or penalties based on its performance.

The AI learns through experience, improving its policy $\pi(s) \rightarrow a$ overtime. Its objective is to maximize its cumulative reward by learning when to attack, dodge, reload, or exploit predictable player patterns. The agent observes the current state of the game, selects an action, and receives a reward depending on whether it succeeded or failed.

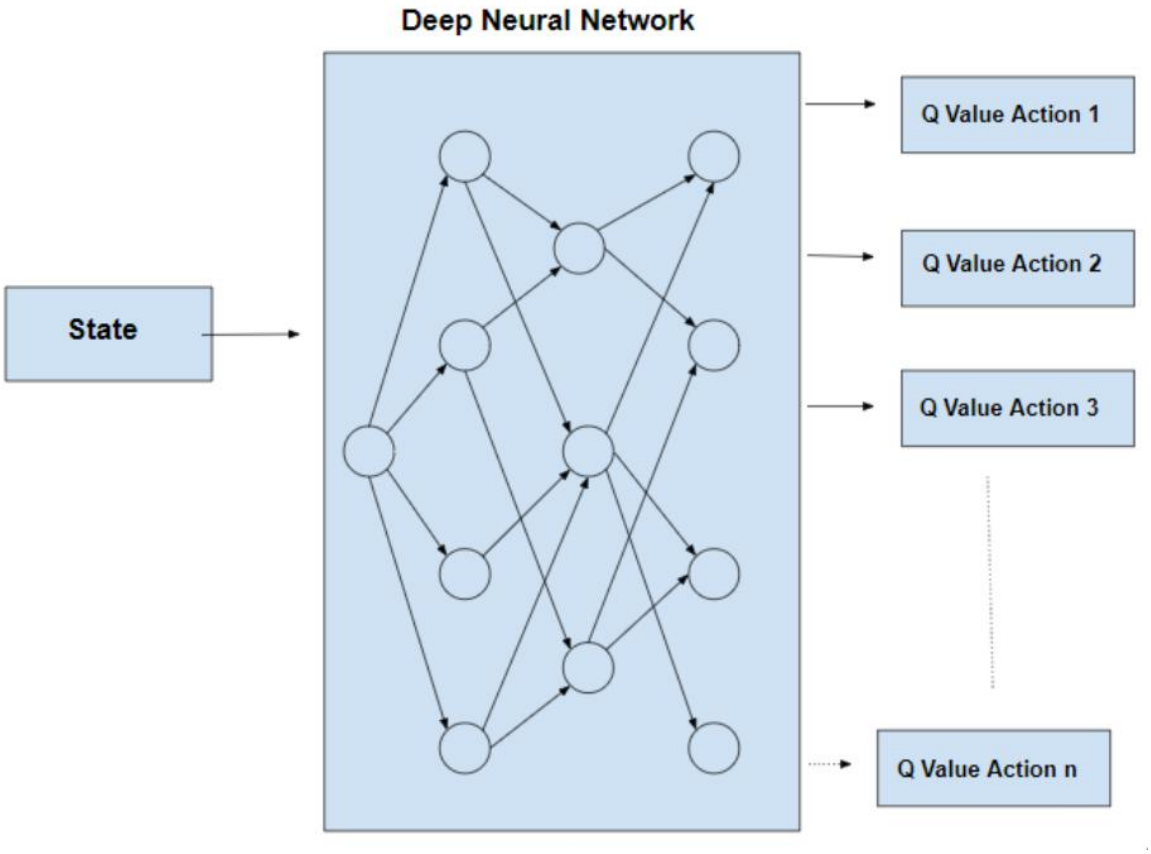
For example, if the AI recognizes a repeated jump-shoot pattern and counters it, it receives a positive reward. If it misses or takes unnecessary damage, it is penalized. Over thousands of gameplay rounds, this system enables the AI to adapt, evolve, and become increasingly difficult to defeat creating a dynamic and engaging opponent for the player.




**2.17 : BASIC ALGORITHM USED FOR RL**

**DEEP Q-LEARNING IN REINFORCEMENT LEARNING :**

We use Deep Q-Learning to train the AI enemy in PAINT PROTOCOL. The model observes the game state and selects actions based on Q-values predicted by a neural network. Through experience replay, target networks, and a carefully designed reward system, the enemy learns to adapt to the player's strategies. This enables dynamic, evolving gameplay where the AI not only mimics but improves becoming an intelligent, strategic opponent.




**2.18 : ARCHITECTURE OF DEEP Q-NEWORK**

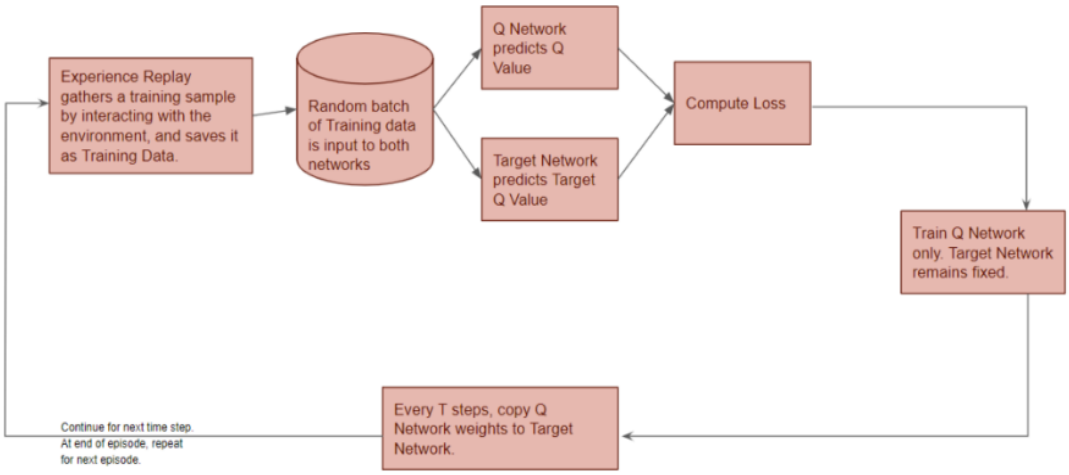
**Key Challenges Addressed by Deep Q-Learning :**

1. High-Dimensional State Spaces: Traditional Q-Learning uses a table to store values but this becomes impossible when there are too many situations. Neural networks can understand and work with many different situations at once so they are better for complex problems.
2. Continuous Input Data: Real-world problems often have continuous data like video images. Neural networks are good at handling this kind of information.

3. Scalability: Deep learning helps Q-Learning grow and handle bigger, harder tasks that regular Q-Learning couldn't solve before.

Deep Q-Learning (DQN) is like giving your AI enemy a brain + memory + a scoreboard. It combines the power of Q-learning (classic reinforcement learning) with deep neural networks, allowing the AI to learn from trial and error and improve over time just like a human player would but smartly.

- ❖ In our game, DQN helps the enemy:
- ❖ Learn when to shoot, dodge, or reload.
- ❖ Predict when the player is about to do something dumb (like repeat a move).
- ❖ Adapt based on rewards it gets for smart decisions.



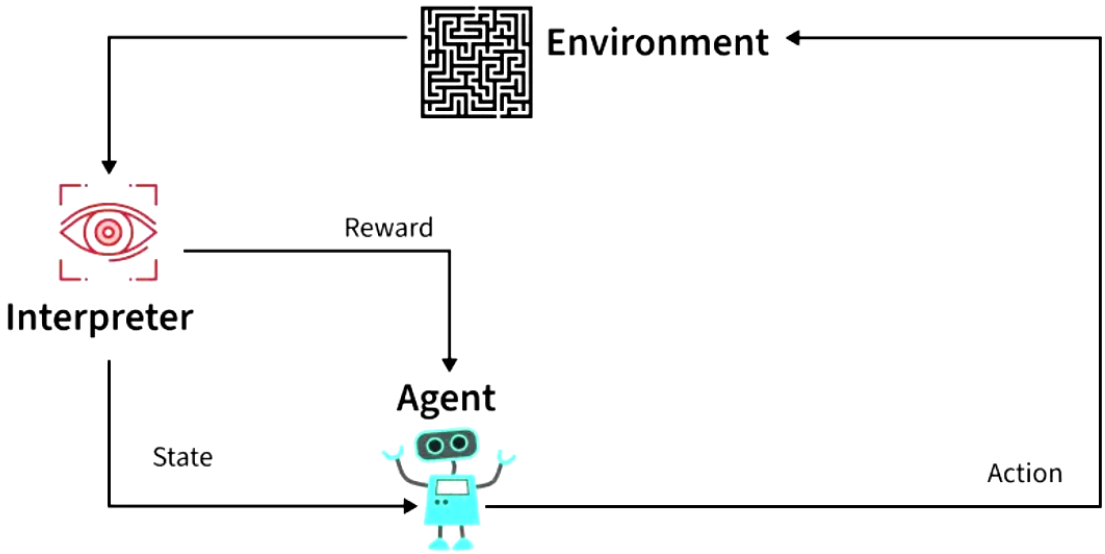

**2.19 : HIGH LEVEL DQN WORKFLOW**

**REWARD SYSTEM IN REINFORCEMENT LEARNING :**

Reinforcement Learning revolves around the idea that an agent (the learner or decision-maker) interacts with an environment to achieve a goal. The agent performs actions and receives feedback to optimize its decision-making over time.

- ➢ **Agent:** The decision-maker that performs actions.
- ➢ **Environment:** The world or system in which the agent operates.

- **State:** The situation or condition the agent is currently in.
- **Action:** The possible moves or decisions the agent can make.
- **Reward:** The feedback or result from the environment based on the agent's action.




**2.20 : REWARD SYSTEM IN REINFORCEMENT LEARNING**

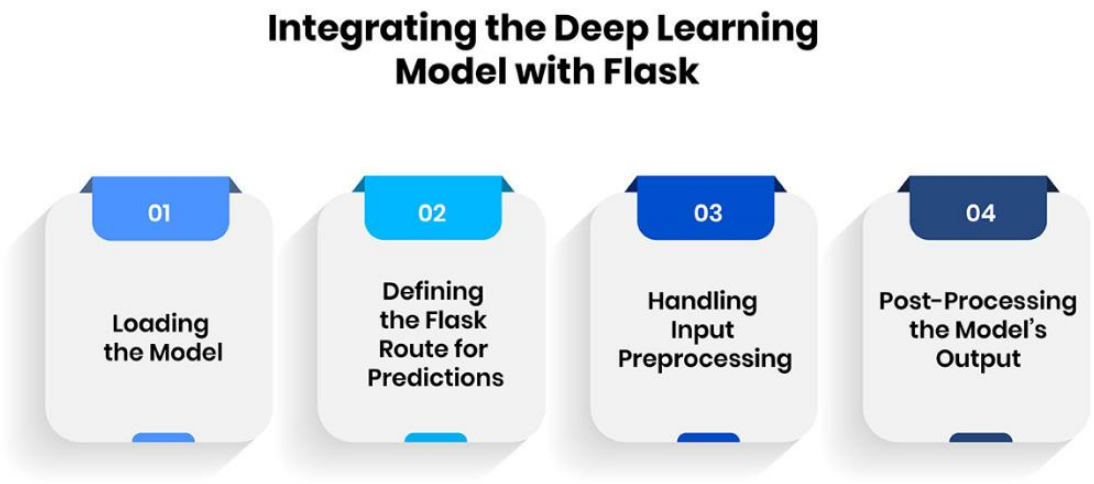
The RL process involves an agent performing actions in an environment, receiving rewards or penalties based on those actions, and adjusting its behavior accordingly. This loop helps the agent improve its decision-making over time to maximize the cumulative reward.

**Example Rewards :**

- +1 for punishing habits of player.
- +0.5 for hits.
- –0.5 for misses.
- –1 for taking a hits.

# D. MODEL DEPLOYMENT (FINAL STEPS)

For model deployment, we adopt a real-time inference strategy using Flask. GODOT sends the current game state to a Flask API, which returns the AI's predicted action based on either the Behavioral Cloning model or the Reinforcement Learning model. This ensures the AI enemy responds immediately to live gameplay conditions, making it adaptive and interactive.




The model deployment architecture uses a client-server approach where the Paint Protocol game acts as the client, and a Flask-based API server hosts the trained AI models. During gameplay, Godot sends the current state of the game to the Flask server via HTTP. Flask processes the input, runs it through the AI model, and returns a predicted action that the enemy executes in-game. The entire server can be containerized using Docker, ensuring easy deployment and scalability. This real-time loop forms the core of our adaptive enemy behavior system.

**FOR EXAMPLE :**

1. **HTTP REQQUEST FROM GODOT TO FLASK**

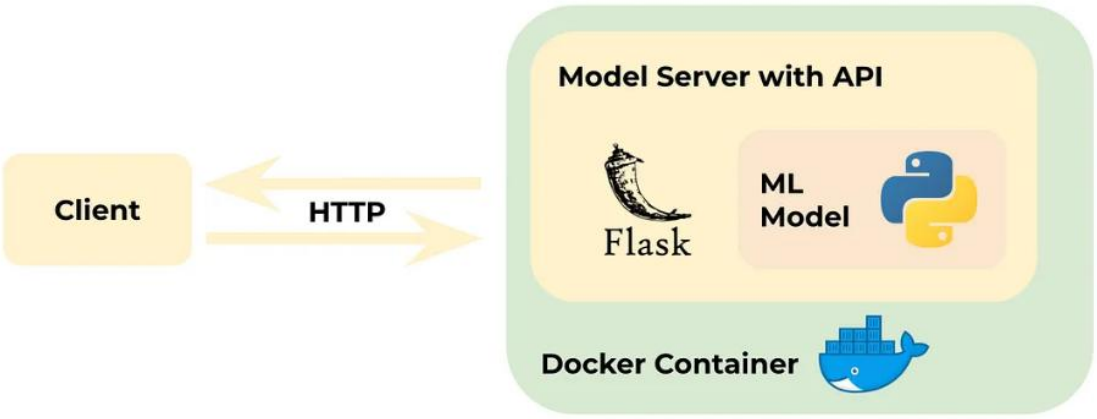
```
POST /predict HTTP/1.1
Host: localhost:5000
Content-Type: application/json

{
```

```
 "player_x": 14.3,
 "player_y": 7.8,
 "enemy_x": 10.2,
 "enemy_y": 5.6,
 "enemy_health": 60,
 "ammo_left": 3,
 "weapon_type": "burst",
 "pattern_flag": 1,
 "is_crouching": false
}
```


2. **THE FLASK RECIEVES THE DATA AND SENDS IT TO THE AI MODEL FOR PROCESSING. ONCE ITS DONE THE AI SENDS ITS FINAL ACTION AS :**


```
{
 "action": "dodge_left"
}
```


**2.21 : BASIC FLASK WORKFLOW**

- Here the client itself is the GODOT ENGINE.
- To ensure easy and portable deployment of the AI model server, the entire Flask + model setup can optionally be packaged into a Docker container. This approach provides environment consistency across machines, making the system scalable, reproducible, and deployment-ready even on cloud platforms or remote servers. While Docker is not essential for local testing, it is recommended for long-term integration.

Section 3

# WORKFLOW :

## 3.1 Week 1

Get comfortable with the Godot engine and scripting. Study the fundamentals of AI, machine learning, and game AI design.

## 3.2 Week 2

Understand behavior logging, pattern detection, and n-gram analysis. Explore how data is structured for AI learning in games.

## 3.3 Week 3

Add gameplay data collection in Godot to track player movement and actions. Save it in a structured format like JSON or CSV for future use.

## 3.4 Week 4

Analyze player logs to identify repeated action patterns. Prepare labeled datasets for training the Behavioral Cloning model.

## 3.5 Week 5

Use supervised learning to mimic player behavior using an MLP model. Save the trained model for real-time inference during gameplay.

## 3.6 Week 6

Design the reward structure and environment setup for DQN. Begin training the AI to react and improve through trial-and-error.

### 3.7 Week 7

Build a real-time Flask server to serve the trained models.This connects the game to the AI for live decision-making.

### 3.8 Week 8

Connect Godot with Flask using HTTP to enable real-time AI control. Test in-game behavior using both BC and RL models.

### 3.9 Week 9

Evaluate enemy adaptiveness, adjust models or rewards if needed. Fix errors.

“ENEMY AI IN THE GAME LEARNS AND ADAPTS…….

TILL THAT TIME

PROJECT X MENTORS FIRING UNEXPECTED TOUGH QUESTIONS FASTER THAN MY PAINTBALL GUN….

AND THE ENEMY WINS THE GAME…”

# REFERENCES :

## A. PART


- ✓ **https://eudl.eu/pdf/10.4108/eai.5-11-2015.150615**
- ✓ **https://www.researchgate.net/publication/368590412_Analyzing_Strengths_and_Weaknesses_of_Modern_Game_Engines**
- ✓ **https://www.intechopen.com/chapters/57425**
- ✓ **https://delftswa.gitbooks.io/desosa2018/content/godot/chapter.html**
- ✓ **https://docs.godotengine.org/en/stable/getting_started/step_by_step/nodes_and_scenes.html**
- ✓ **https://docs.godotengine.org/en/stable/tutorials/animation/introduction.html**
- ✓ **https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html**


## B. PART


- ✓ https://github.com/Burloe/GoLogger
- ✓ https://www.loggly.com/use-cases/json-logging-best-practices/


## C. PART


- ✓ https://en.wikipedia.org/wiki/Multilayer_perceptron
- ✓ https://www.researchgate.net/figure/Process-of-Behavioral-cloning_fig4_383950886
- ✓ https://michael-fuchs-python.netlify.app/2021/02/03/nn-multi-layer-perceptron-classifier-mlpclassifier/#introduction
- ✓ https://blog.tensorflow.org/2019/01/whats-coming-in-tensorflow-2-0.html
- ✓ https://www.turing.com/kb/building-neural-network-in-tensorflow
- ✓ https://www.geeksforgeeks.org/machine-learning/a-brief-introduction-to-proximal-policy-optimization/


## D. PART


- ✓ https://www.gameaipro.com/GameAIPro/GameAIPro_Chapter48_Implementing_N-Grams_for_Player_Prediction_Proceedural_Generation_and_Stylized_AI.pdf?utm_source=chatgpt.com
- ✓ https://funnel.io/blog/n-gram-analysis
- ✓ https://github.com/tensorflow/agents/tree/v0.19.0


## E. PART


- ✓ https://www.geeksforgeeks.org/machine-learning/what-is-reinforcement-learning/

- ✓ https://towardsdatascience.com/reinforcement-learning-explained-visually-part-5-deep-q-networks-step-by-step-5a5317197f4b/
- ✓ https://www.geeksforgeeks.org/deep-q-learning/

## F. PART

- ✓ https://www.geeksforgeeks.org/machine-learning/deploy-machine-learning-model-using-flask/
- ✓ https://engineering.rappi.com/serve-your-first-model-with-scikit-learn-flask-docker-df95efbbd35e

Appendix A

# TASK 1 : LOST IN TRANSLATION

**LINK :** https://github.com/AvanishSalunke/LOST-IN-TRANSLATION.-AVANISH

## A.1 : CODE

```
# this are the test cases
# 1 - In April 2023, Sundar Pichai did announce that Google would be
launehing a new AI product namcd Gemini. Barack Obama also gave a speech
at Harvard University, cmphasizing the role of technology in modern
education.
# 2 - Project X is an exclusive elub at Veermata Jijabai Technological
Institute, Mumbai, mcant to 5erve as a healthy environment for 5tudents to
learn from each other and grow together. Through the guidance of their
mcntors these 5tudents are able to complete daunting tasks in a relatively
short time frame, gaining significant exposure and knowledge in their
domain of choice.
# 3 - I will be eompleting my BTech dcgree in Mechanical Engineering from
VJTI in 2028
# 4 - However the faet that Project X is always going to TOP is insane.

##########################################################################
################################################

import nltk
import spacy
import nltk
nltk.download('punkt_tab')
nltk.download('words')
nltk.download('wordnet')
from nltk.corpus import words, wordnet
english_words = set(words.words())
def is_english_word(word):
    return (word in english_words) or (len(wordnet.synsets(word)) > 0)
### the reason for making the or in this one ,,,,, is the 5tudnet case ,
it changes to students n its not in word database
```

```
text = input("Enter text: ")
tokens = nltk.word_tokenize(text)

##############################################################################
###############################################

nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
protected_words = set()
for ent in doc.ents:
    if ent.label_ in ['PERSON', 'ORG', 'GPE', 'DATE']:
        for word in ent.text.split():
            if word.isalpha():
                protected_words.add(word)

##############################################################################
###############################################

mapping = {
    '0': 'o',
    '1': 'l',
    '5': 's',
    '6': 'g',
    '8': 'b',
    'b': 'h',
    'c': 'e',
    'h': 'b',
    'l': 'i',
    'i': 'l',
    'e': 'c',
    'f': 't',
    'a': 'o',
    'm': 'n',
    'n': 'm'
}
final_tokens = []

##############################################################################
###############################################

for token in tokens:
    if token in protected_words:
        final_tokens.append(token)
        continue
```

```
    token_lower = token.lower()
    if is_english_word(token_lower):
        final_tokens.append(token)
        continue
    corrected = None
    for i in range(len(token_lower)):
        ch = token_lower[i]
        if ch in mapping:
            candidate = token_lower[:i] + mapping[ch] + token_lower[i+1:]
            if is_english_word(candidate):
                corrected = candidate
                break
    if corrected is not None:
        final_tokens.append(corrected)
    else:
        final_tokens.append(token)
final_text = ' '.join(final_tokens)

##########################################################################
#######################################

import re
discourse_markers = ['However', 'Therefore', 'Through', 'Meanwhile',
'Although', 'Moreover']
for marker in discourse_markers:
    pattern = rf'\b{marker}\b'
    final_text = re.sub(pattern, marker + ',', final_text)

##########################################################################
#########################################

print(final_text)
doc = nlp(final_text)
target_labels = {"PERSON", "ORG", "GPE"}
entities = []
for ent in doc.ents:
    if ent.label_ in target_labels:
        entities.append(ent.text)
print(entities)
```

## A.2 : OUTPUT

```
Test Case 1:
Fixed Text:
 In April 2023 , Sundar Pichai did announce that Google would be launching a new AI product named Gemini . Barack Obama also gave a speech
 at Harvard University , emphasizing the role of technology in modern education .
Named Entities: ['Sundar Pichai', 'Google', 'AI', 'Gemini', 'Barack Obama', 'Harvard University']

Test Case 2:
Fixed Text:
 Project X is an exclusive club at Veermata Jijabai Technological Institute , Mumbai , meant to serve as a healthy environment for student
 to learn from each other and grow together . Through, the guidance of their mentors these students are able to complete daunting tasks in
 relatively short time frame , gaining significant exposure and knowledge in their domain of choice .
Named Entities: ['Project X', 'Veermata Jijabai Technological Institute', 'Mumbai']

Test Case 3:
Fixed Text:
 I will be completing my BTech degree in Mechanical Engineering from VJTI in 2028
Named Entities: ['BTech', 'Mechanical Engineering', 'VJTI']

Test Case 4:
Fixed Text:
 However, the fact that Project X is always going to TOP is insane .
Named Entities: []

[Done] exited with code=0 in 9.476 seconds
```

Appendix B

# TASK 2 : PROJECTX APTITUDE TEST

**LINK :** https://github.com/AvanishSalunke/PROJECTX_APTITUDE_TEST.AVANISH

### B.2 : CODE

```
import cv2
import numpy as np
raghav = cv2.imread("Raghav.jpg", cv2.IMREAD_GRAYSCALE)
ghanshyam = cv2.imread("Ganshyam.jpg", cv2.IMREAD_GRAYSCALE)
bhaskar = cv2.imread("Bhaskar.jpg", cv2.IMREAD_GRAYSCALE)
merged = cv2.merge((bhaskar, ghanshyam, raghav))
gray = cv2.cvtColor(merged, cv2.COLOR_BGR2GRAY)
_, binary = cv2.threshold(gray, 1, 255, cv2.THRESH_BINARY)
# cv2.imshow("All Shapes in White", binary)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
#now in my laptop the image was coming veryy big out of the screen so we
might have to scale it down but anyways for the answer part it was not
needed
height, width = binary.shape
#the number of columns shd be 9 ,,, but when the program was executed it
gave wrong answers becausee there is some black space below which gave
error sooo just manipulated it.......
total_rows = 10
total_cols = 5
row_height = height // total_rows
col_width = width // total_cols
answers = []
for row in range(1, total_rows-1): #### the reasons for 1 to this is
because i dont want to scan the labels and the darkspace below
    max_white = -1
    selected_col = -1
    for col in range(1, total_cols):
        y1 = row * row_height
        y2 = (row + 1) * row_height
```

```
            x1 = col * col_width
            x2 = (col + 1) * col_width
            cell = binary[y1:y2, x1:x2]
            white_pixels = cv2.countNonZero(cell)
            if white_pixels > max_white:
                max_white = white_pixels
                selected_col = col - 1
        answer_letter = chr(ord('A') + selected_col)
        answers.append(f"{row} - {answer_letter}")
for ans in answers:
    print(ans)
```

**B.2 : OUTPUT**

```
1 - D
2 - A
3 - B
4 - C
5 - C
6 - B
7 - B
8 - D

[Done] exited with code=0 in 1.494 seconds
```

Appendix C

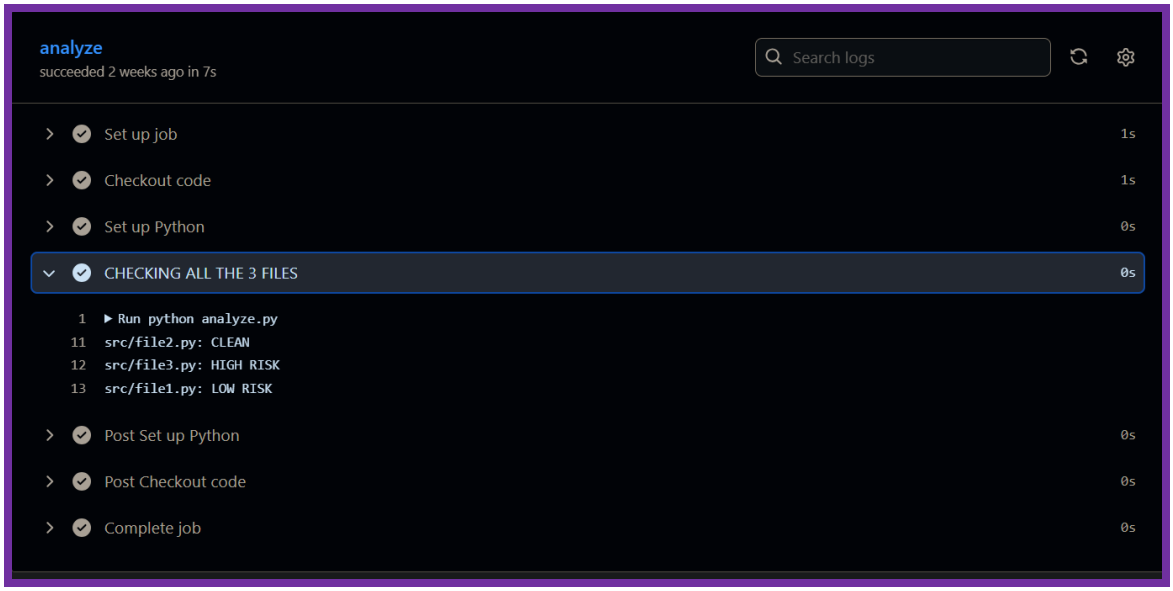
# TASK 3 : SHED YOUR BUGS

**LINK :** https://github.com/AvanishSalunke/SHED_UR_BUGS.AVANISH

### C.1 : CODE

```
import os
forbidden_keywords = ['print', 'eval', 'exec']
for files in os.listdir("src"):
    path = os.path.join("src", files)
    violation_count = 0
    can_i_break_program = False
    with open(path, "r") as file:
        lines = file.readlines()
    for line in lines:
        if len(line.rstrip()) > 80:
            violation_count += 1
        code_only = line.split('#')[0]
        # this anyways finds both the fulll type comment nn inline comment
so its better other than finding the types of comment by two diff wayss
        for word in forbidden_keywords:
                if word in code_only:
                    can_i_break_program = True
                    violation_count += 1
                    break
        if line.count('"') % 2 != 0 or line.count("'")%2 != 0 :
                violation_count += 1
    if violation_count == 0:
        status = "CLEAN"
    elif can_i_break_program or violation_count > 5:
        status = "HIGH RISK"
    else:
        status = "LOW RISK"
    print(f"src/{files}: {status}")
```

## C.3 : OUTPUT


analyze
succeeded 2 weeks ago in 7s
Search logs
Set up job 1s
Checkout code 1s
Set up Python 0s
CHECKING ALL THE 3 FILES 0s
1 ▶ Run python analyze.py
11 src/file2.py: CLEAN
12 src/file3.py: HIGH RISK
13 src/file1.py: LOW RISK
Post Set up Python 0s
Post Checkout code 0s
Complete job 0s