AI Project Report

Group Members:
Ayush (12340420), Ajay (12340580), Kishor (12341210), Pobitro (12341580)
Project : Pokemon battle simulator

Overview:
This project implements a simplified but functional Pokémon battle simulator. It features matches between a human Player vs AI/computer player. Both parties start out with 3 randomly generated pokemon. To give a bit of freedom to the player, the game generates multiple sets of 3 pokemon amongst which the player can choose his set.

The objective of the game is straightforward: strategically use your Pokémon and their moves to defeat all three opposing Pokémon before your own team is knocked out. The battle continues until one side has no remaining usable Pokémon, at which point the other side is declared the winner.

Mechanics:
The actual Pokemon battle simulator consists of a wide variety of mechanics such as abilities, items, status conditions, weather effects, terrains, multi-turn moves, switching strategies, and more. Since it is not possible to implement all of these within a month's timeframe, we focused on the most fundamental and impactful mechanics, which keeps the game still playable and fun.

Our main mechanics are such:

Type System
Each pokemons has a *type.* Each type has its respective other types which it is effective against and is weak against. Weakness or effectiveness basically gives it a boost against the opponent pokemon on how much damage it can do or how much incoming damage would be reduced by. Types can be applicable to a certain pokemon or its moves, with move type deciding its attack potency and pokemon type deciding its defense effectiveness. Pokémon can have dual types, meaning a single incoming attack may be affected by two different type matchups simultaneously. Moves, however, can only have one type and do not depend on the pokemons type, for example a water type pokemon ironically may possess a fire type move. (In the actual pokemon there are a select few examples of dual type moves but since they are very less we have kept move types to single)

Type effectiveness can lead to 3 outcomes:
Super effective moves deal increased damage.
Not very effective moves deal reduced damage.
Neutral interactions deal normal damage.

For which type is effective against which it is too much to explain in this report. For that you may google that as there are 18 types in total and each having a weakness and effectiveness pair.

Speed and Turn order
Every Pokémon has a speed stat. During each turn, the Pokémon with the higher speed value gets to attack first. If speeds are equal, a random tie-breaker can determine who moves first. This makes speed an essential part of the battle strategy, especially when predicting knockouts. Opponent pokemon speeds are not displayed to

Approach:
To make the gameplay challenging we have used alpha beta pruning using minimax tree function to determine the Ai's moves. The heuristic our function relies on consists of multiple factors.

Heuristic:
Type effectiveness: The AI checks how effective its possible moves would be against the player's current Pokémon and prefers super effective moves when available.
Incoming effectiveness: The heuristic also evaluates how dangerous the player's potential counterattacks could be.
HP difference and remaining HP ratio: The AI considers how much health both sides have, rewarding states where the AI has more HP remaining.
Knockout potential: Special cases are prioritized, such as when the AI can finish off the opponent immediately or when the opponent could potentially knock out the AI on the next turn.
Move selection and minimization of risk: By comparing damage outputs and risks from each possible move, the heuristic determines an optimal or near-optimal action.

Project navigation:
The project is split into multiple folders with each containing a core part of the application.

Assets folder: The **Assets** folder contains all graphical resources used in the simulator.
This includes: sprites and images of all pokemons and backgrounds used by the simulator
Data folder: This contains all the data and statistics regarding a pokemon like what type it is, lvl, attack power, type effectiveness etc. all of this is in the bulbapedia_data.csv file.
Core folder: This consists of the main logic regarding, deriving information about a pokemon. The main file is the battle state file. This tells us the current state regarding a pokemon and gives us all available and possible data. The other files are helper functions which are used by battlestate or other files. Pokemon.py file gives information regarding pokemon and deals with actions on the pokemon like taking damage or attacking or getting stats etc. Move.py gives information regarding moves and also contains functions to calculate damage which is used in heuristics. Type effectiveness just gives a flow cycle and multipliers as to which type should get what boost/nerf etc. ability.py updates battlestate.

AI folder: This folder contains the computer player logic. Heuristic.py contains the heuristic used to measure and calculate effectiveness of the next move. Minimax agent is the main alpha beta pruning minimax file which returns the move the ai should make. (if you are wondering what mcts is we tried experimenting with Monte Carlo tree search however we have decided to stick to alpha beta pruning)

Engine folder: this folder contains all files pertaining to the application and making it run. It contains the starting logic and how the screen should be, how the men should be, what to execute when what is clicked, keeping the game looped until it ends etc. what to display when the game ends, what to display when the game is running atc.

Our main file is main_ds.py (NOT main.py) so run that to start. That file basically starts the entire process.

Project Achievement Summary

Throughout the development of the Pokémon Battle Simulator, our team successfully implemented a functional and engaging turn-based battle system supported by an intelligent AI opponent. The project demonstrates our ability to combine game mechanics, algorithmic decision-making, and structured software design within a limited timeline.