**Project Report**

**Project Title:** Dynamic Fighter Swapping Mechanic in Street Fighter (Warriors Showdown)

**Submitted By:** Baasim Ahmed (22K-5029)

Danish Khan (22K-5107)

Muhammad Fawaz (22K-5053)

**Course:** AI  
**Instructor:** Alishba Subhani  
**Submission Date:** 11th May 2025

**1. Executive Summary**

* Project Overview:

This project involved the development of "Warriors Showdown," a 2D fighting game using the Pygame library. The core objective was to create a game environment capable of supporting multi-character teams with character switching mechanics, serving as a platform for future AI agent development. Key modifications to a conventional fighting game structure include the implementation of a team-based system, a character switching feature with a cooldown, a main menu with game state management (Single Player, Multiplayer, Quit), and a basic placeholder AI for the single-player mode. The project successfully established a functional framework for both local human-vs-human multiplayer and a human-vs-AI mode.

**2. Introduction**

* Background:

Fighting games are a popular genre that presents interesting challenges for AI development due to their real-time nature, complex state spaces, and the need for reactive and predictive decision-making. "Warriors Showdown" was selected as the project basis to explore these challenges, specifically introducing the novel element of character switching within a match. This innovation adds a strategic layer, requiring players (and eventually the AI) to manage multiple character resources and make tactical decisions about when to switch fighters.

* Objectives of the Project:

The primary objectives of this project were to:

* + Develop a functional 2D fighting game engine using Pygame.
  + Implement a multi-character team system for each player.
  + Create a character switching mechanic with a defined cooldown period.
  + Design and implement a main menu for game state navigation (Single Player, Multiplayer, Quit).
  + Establish a local human-vs-human multiplayer mode.
  + Integrate a basic placeholder AI for a human-vs-AI single-player mode, setting the stage for advanced AI training.
  + Ensure a stable game loop and basic fighting interactions (movement, attacks, health).

**3. Game Description**

* Original Game Rules (Conceptual Basis):

"Warriors Showdown" draws inspiration from traditional 2D fighting games. The basic rules involve two characters facing off, using various moves (punches, kicks, special abilities) to reduce the opponent's health to zero. Players typically move left and right, jump, block, and attack. The first player to deplete the opponent's health bar wins the round, and winning a set number of rounds wins the match.

* Innovations and Modifications:

This project introduces significant modifications to the conventional fighting game structure:

* + **Team-Based Combat:** Players select a team of multiple fighters (currently two in the implementation) rather than just one.
  + **Character Switching:** Players can dynamically switch their active fighter with a benched fighter during the match. This adds strategic depth related to character health management, cooldowns, and exploiting favorable match-ups.
  + **Persistent Health:** Character health persists between switches within a round. A fighter is eliminated from the match when their health reaches zero.
  + **Match Win Condition:** The match is won when a player has no remaining fighters left in their team.
  + **Game States:** Implementation of distinct game states (Menu, Single Player, Multiplayer, Quit) for structured navigation.

**4. AI Approach and Methodology**

* AI Techniques Used:

For the single-player mode, the project currently utilizes a placeholder AI. This AI does not employ complex learning algorithms like Reinforcement Learning or search algorithms like Minimax. It follows a very simple set of hardcoded rules based on the current game state (e.g., move towards the opponent, attack when close, attempt to jump occasionally, switch when health is low).

* Algorithm and Heuristic Design:

The current AI's "logic" is based on simple conditional statements (if-then rules).

* + **Movement:** Move horizontally towards the opponent if not within a certain distance threshold. Stop moving when close.
  + **Attacking:** Attempt an attack if within a defined attack range and the attack cooldown is ready. Alternates between attack types 1 and 2 based on a simple timer check.
  + **Jumping:** Has a small random chance to jump if not already jumping or attacking.
  + **Switching:** Attempts to switch to the benched character if the active character's health drops below a certain threshold, the benched character is alive, and the switch cooldown is ready.

This placeholder AI is intentionally simple and serves only to make the AI-controlled character perform some basic actions. The design is meant to be replaced by a more sophisticated AI agent developed using advanced techniques.

* AI Performance Evaluation:

Formal AI performance evaluation (win rate, decision time, accuracy against human players or other AIs) has not yet been conducted as the current AI is a placeholder. Future work involving a trained AI agent would require rigorous evaluation against human players or benchmark AIs to assess its effectiveness and strategic capabilities, particularly in utilizing the character switching mechanic.

**5. Game Mechanics and Rules**

* **Modified Game Rules:**
  + Each player controls a team of two fighters.
  + Players can switch their active fighter with their benched fighter using a dedicated key/input.
  + A cooldown period is enforced after each character switch, preventing rapid, continuous switching.
  + Fighter health is persistent throughout a round.
  + A fighter is eliminated from the match when their health reaches zero.
  + The round ends when one player's active fighter is defeated and they have no remaining benched fighters.
  + The match is won when a player has no alive fighters left in their team across all rounds.
  + Rounds are won by eliminating all of the opponent's fighters. The first player to win 2 rounds wins the match.
* Turn-based Mechanics:

"Warriors Showdown" is a real-time fighting game, not turn-based. Both players can move, attack, and switch characters simultaneously, limited by cooldowns and character states (e.g., not being able to attack while being hit). The game loop continuously updates character positions, animations, health, and checks for collisions and inputs.

* **Winning Conditions:**
  + **Round Win:** A player wins a round by defeating all of the opponent's fighters (reducing their health to zero).
  + **Match Win:** A player wins the match by winning a predetermined number of rounds (currently set to 2 rounds).

**6. Implementation and Development**

* Development Process:

The project followed an iterative development process. Initially, core fighting mechanics (movement, basic attacks) were implemented for single characters. The system was then extended to handle multiple characters and the switching mechanic. A state machine was introduced to manage the different game screens (menu, gameplay). Input handling was implemented for both single-player (human + AI placeholder) and local multiplayer (two human players). Asset loading and basic rendering were integrated using Pygame's functionalities.

* **Programming Languages and Tools:**
  + **Programming Language:** Python
  + **Libraries:** Pygame
  + **Tools:** Visual Studio Code (or similar IDE), Command Line Interface for running the game. (Optionally, Git for version control).
* **Challenges Encountered:**
  + **Animation Management:** Synchronizing character animations with actions (movement, attacks, hits) and ensuring smooth transitions.
  + **Collision Detection:** Implementing accurate collision detection for attacks and character interactions.
  + **Character Switching Logic:** Ensuring the switching mechanic worked correctly, handling cooldowns, and transferring state (position, velocity) between fighters.
  + **Game State Management:** Structuring the code to transition smoothly between the menu and gameplay states.
  + **AI Placeholder Integration:** Setting up the game loop to accept actions from either human input or the AI placeholder based on the game state.
  + **Asset Loading:** Ensuring all necessary assets were correctly located and loaded by Pygame, handling potential errors gracefully.

**7. Team Contributions**

* [Danish]: Responsible for [Core Fighter Class Implementation, Animation Logic].
* [Baasim]: Responsible for [Character Switching Mechanics, Game State Management].
* [Fawaz]: Responsible for [Menu Implementation, Input Handling, Asset Integration].
* [Baasim]: Responsible for [Basic AI Placeholder Logic, Bug Fixing].

**8. Results and Discussion**

* AI Performance:

As the current AI is a simple placeholder, its performance is limited. It can perform basic actions and will move towards the opponent and attack, but it lacks strategic depth, predictive capabilities, or effective use of character switching. It serves its purpose as a test opponent for the single-player mode's framework but would not provide a challenging experience for a skilled human player. The primary "result" in the AI domain so far is the successful creation of an environment where a more advanced AI can be integrated and trained.

* Game Functionality:

The core game mechanics, including movement, attacks, health, and character switching, are functional in both Single Player (with basic AI) and Local Multiplayer modes. The menu system allows players to navigate between game modes. The round and match win conditions are tracked.

* Discussion:

The project successfully built a solid foundation for a multi-character fighting game with character switching. The most significant remaining AI task is the development and training of a sophisticated AI agent that can effectively utilize the game's mechanics, particularly the character switching, to provide a challenging opponent in single-player mode. This would involve exploring advanced AI techniques like Reinforcement Learning and designing appropriate reward functions and state representations.

**9. References**

* Pygame Documentation: https://www.pygame.org/docs/