## 📄 ****Semester Project Proposal****

### ****Project Title****:

**Tic-Tac-Five: An Intelligent 5x5 Tic-Tac-Toe Game with AI Modes and Comparative Performance Metrics**

### ****Group Members****:

* Muhammad Abbas (23k0068)
* Hamza Sheikh (23k0060)
* Muhammad Sabeeh (23k0002)
* Zaid Bin Naveed (23k0028)

### ****Project Description****:

This project involves the development of a 5x5 Tic-Tac-Toe game—a strategic twist on the classic 3x3 version—requiring four marks in a row (horizontally, vertically, or diagonally) to win. The application features two intelligent game modes and a user-friendly GUI built using Python’s Tkinter.

### ****Gameplay Modes****:

#### 1. ****AI vs AI Mode****

In this mode, two advanced AI models compete:

* **Minimax with Alpha-Beta Pruning**
* **Monte Carlo Tree Search (MCTS)**

Each game automatically plays out, and after each match, the following performance metrics are displayed:

* Number of nodes explored
* Average decision time per move
* Memory usage
* Win/loss/draw outcomes across multiple simulations

This setup allows for a **comparative performance analysis** between the two AI approaches under identical conditions.

#### 2. ****Human vs AI Mode****

The user competes against the Minimax AI with adjustable difficulty:

* **Easy**: Random or shallow-depth Minimax
* **Medium**: Minimax with limited depth and basic heuristic
* **Hard**: Full Minimax with Alpha-Beta Pruning and refined heuristic

The difficulty setting dynamically adjusts the AI’s evaluation depth and heuristic precision to ensure an engaging challenge for players of all skill levels.

### ****Search Algorithm & AI Design****:

* **Minimax Algorithm**: Evaluates all game states to find optimal moves using a heuristic based on potential sequences of 2, 3, or 4 symbols.
* **Alpha-Beta Pruning**: Reduces unnecessary branches to optimize computation time.
* **Monte Carlo Tree Search**: Provides a probabilistic approach to decision-making through randomized simulations and backpropagation.
* **Dynamic Heuristic Evaluation**: Adapted to difficulty level and AI type for accurate board assessments.

### ****User Interface Design****:

* Developed using **Tkinter**
* Clear 5x5 grid layout with buttons
* Real-time move updates and audio/visual cues
* Scoreboard and status display
* Mode and difficulty selection menu
* Game restart and exit controls with user confirmation

### ****Comparative Analysis & Performance Evaluation****:

We will compare:

* **Minimax vs Alpha-Beta Pruning vs MCTS** (in AI vs AI mode)
* **Minimax difficulty levels** (in Human vs AI mode)

Metrics:

* Nodes explored
* Average decision time
* Memory consumption
* Win/draw/loss ratio for both AIs
* Heuristic accuracy and decision quality

### ****Complexity Analysis****:

* Time and space complexity for:
  + Basic Minimax
  + Alpha-Beta optimized Minimax
  + Monte Carlo Tree Search
* Visualization of performance degradation/improvement with:
  + Increased board size
  + Difficulty levels
  + Number of simulations (in MCTS)

### ****Documentation****:

Final submission will include:

* Complete and modular source code with documentation
* GUI and AI module explanations
* Comparative graphs and tables
* A formal report with experimental results and analysis
* Optional presentation or demo video

### ****Why It Fits the Project Requirements****:

| **Requirement** | **Met?** | **Explanation** |
| --- | --- | --- |
| AI-driven Decision Making | ✅ | Minimax, Alpha-Beta Pruning, MCTS |
| Search Space Expansion | ✅ | 5x5 board with a 4-in-a-row win condition |
| Efficient Search Implementation | ✅ | Optimization via pruning and simulation-based MCTS |
| Intuitive & Engaging UI | ✅ | Interactive Tkinter-based interface |
| Comparative Analysis | ✅ | Head-to-head AI comparisons with metrics |
| Well-Documented Code & Report | ✅ | Structured documentation and final report |

Let me know if you'd like this saved into an updated .docx file or need help implementing any of the features.