**Project Overview**

You need to develop a C-based Tic Tac Toe game with AI/ML capabilities for IoT platforms, targeting cognitive learning for children.

**Deadline:** November 25, 2025, 23:59  
**Expected Hours:** 15 per team member (60 total)  
**Project Weight:** 35% of final course mark

**PHASE 1: Core Development (Weeks 1-3)**

**Team A: GUI & Two-Player Development**

**Members:** Member 1 & Member 2  
**Timeline:** October 15 - November 4 (3 weeks)

**Member 1: GUI Architecture & Display Systems**

**Primary Focus:**

* Design and implement core GUI framework
* Create game board visualization system
* Develop user interface components
* Handle display rendering and updates

**Specific Tasks:**

* Research and implement GUI library (gnuplot integration or alternative C GUI libraries)
* Create 3x3 game board visualization
* Design symbol rendering system (X and O)
* Implement screen refresh and update mechanisms
* Create menu systems and navigation
* Design winner announcement displays

**Files to Create:**

* gui\_core.c / gui\_core.h - Core GUI framework
* display\_manager.c / display\_manager.h - Display rendering
* board\_renderer.c / board\_renderer.h - Game board visualization
* menu\_system.c / menu\_system.h - Menu and navigation

**Member 2: User Input & Two-Player Game Logic**

**Primary Focus:**

* Implement user input handling systems
* Develop two-player game mechanics
* Create game flow management
* Handle player interaction validation

**Specific Tasks:**

* Implement mouse/keyboard input handling
* Create player turn management system
* Develop move validation logic
* Handle game state transitions
* Implement game reset functionality
* Create player interaction feedback systems

**Files to Create:**

* input\_handler.c / input\_handler.h - Input processing
* two\_player\_logic.c / two\_player\_logic.h - Two-player game mechanics
* game\_state.c / game\_state.h - Game state management
* player\_manager.c / player\_manager.h - Player interaction handling

**Team B: AI & Machine Learning Development**

**Members:** Member 3 & Member 4  
**Timeline:** October 15 - November 4 (3 weeks)

**Member 3: Minimax AI & One-Player Mode**

**Primary Focus:**

* Implement standard Minimax algorithm
* Develop perfect AI player
* Create one-player game mode
* Optimize algorithm for IoT constraints

**Specific Tasks:**

* Research and implement recursive Minimax algorithm
* Create game tree evaluation system
* Develop perfect AI using depth-first search
* Implement move prediction and optimization
* Create difficulty level variations
* Handle game termination scenarios

**Files to Create:**

* minimax\_core.c / minimax\_core.h - Core Minimax implementation
* game\_tree.c / game\_tree.h - Game state tree management
* ai\_perfect.c / ai\_perfect.h - Perfect AI player
* one\_player\_mode.c / one\_player\_mode.h - Single player game logic

**Member 4: Machine Learning Implementation**

**Primary Focus:**

* Implement chosen ML algorithm (Naive Bayes or Linear Regression)
* Process and analyze dataset
* Create imperfect AI for educational purposes
* Generate performance metrics

**Specific Tasks:**

* Process the 958-row dataset (80:20 train/test split)
* Implement chosen ML algorithm in C
* Calculate training and testing accuracy
* Create imperfect AI that allows children to win
* Generate confusion matrices
* Compare ML performance with Minimax

**Files to Create:**

* ml\_algorithm.c / ml\_algorithm.h - ML implementation
* dataset\_processor.c / dataset\_processor.h - Data handling
* ml\_training.c / ml\_training.h - Training functions
* ai\_imperfect.c / ai\_imperfect.h - ML-based AI player

**PHASE 2: Integration, Testing & Documentation (Weeks 4-5)**

**New Role Assignments for Phase 2**

**Timeline:** November 5 - November 25 (3 weeks)

**Member 1: Testing Lead & Cross-Platform Validation**

**Transition Focus:**

* Coordinate comprehensive testing across all components
* Perform cross-platform compatibility testing
* Conduct user experience testing
* Document testing results and bug fixes

**Specific Tasks:**

* Create comprehensive testing framework
* Test GUI compatibility across different systems
* Conduct user testing with friends/family (educational effectiveness)
* Document all bugs and coordinate fixes with other team members
* Ensure IoT platform optimization
* Create testing documentation

**Files to Create:**

* testing\_framework.c / testing\_framework.h - Testing utilities
* gui\_tests.c / gui\_tests.h - GUI-specific tests
* user\_testing\_results.txt - User testing documentation
* platform\_compatibility.txt - Cross-platform testing results

**Member 2: Integration Lead & System Architecture**

**Transition Focus:**

* Integrate all components into cohesive system
* Create main program coordination
* Handle component communication
* Optimize system performance

**Specific Tasks:**

* Develop main program entry point and coordination
* Integrate GUI with AI components
* Create component communication interfaces
* Handle system-wide error management
* Optimize memory usage and performance
* Create build system and deployment procedures

**Files to Create:**

* main.c - Main program entry point
* system\_integration.c / system\_integration.h - Component integration
* error\_handler.c / error\_handler.h - System error management
* Makefile - Build configuration

**Member 3: Performance Analysis & Algorithm Optimization**

**Transition Focus:**

* Analyze and compare algorithm performance
* Optimize AI components for IoT constraints
* Generate performance comparison reports
* Document algorithm effectiveness

**Specific Tasks:**

* Measure Minimax vs ML algorithm performance
* Analyze memory usage and response times
* Create performance comparison charts
* Optimize algorithms for memory constraints
* Document algorithm decision-making processes
* Generate confusion matrices and accuracy reports

**Files to Create:**

* performance\_monitor.c / performance\_monitor.h - Performance measurement
* algorithm\_comparison.c / algorithm\_comparison.h - Algorithm analysis
* confusion\_matrix.c / confusion\_matrix.h - ML analysis tools
* performance\_results.txt - Performance documentation

**Member 4: Documentation Lead & Report Compilation**

**Transition Focus:**

* Coordinate report writing and documentation
* Create presentation materials
* Compile final submission package
* Ensure all deliverables meet requirements

**Specific Tasks:**

* Coordinate report section writing among all team members
* Create and edit final report document
* Develop presentation video content and script
* Compile all source code and documentation
* Ensure submission requirements compliance
* Create demo videos and user guides

**Files to Create:**

* PMReportSessionXGroupXX.pdf - Final report
* README.txt - Setup and usage instructions
* code\_documentation.txt - Code documentation
* Presentation video and demo materials

**Shared Core Files (All Members Contribute in Both Phases)**

**Common Infrastructure:**

* tic\_tac\_toe.h - Main header with game structures and constants
* game\_logic.c / game\_logic.h - Core game rules and validation
* utils.c / utils.h - Utility functions

**Revised Timeline**

**Phase 1: Core Development (3 weeks)**

**Week 1 (Oct 15-21): Foundation**

**Team A (GUI):**

* Member 1: Research GUI libraries, create interface mockup
* Member 2: Design input handling system, create two-player logic outline

**Team B (AI/ML):**

* Member 3: Research Minimax algorithm, create implementation plan
* Member 4: Analyze dataset, choose ML algorithm, create processing plan

**Deliverable:** Technical specifications and pseudocode for all components

**Week 2 (Oct 22-28): Core Implementation**

**Team A:**

* Member 1: Implement basic GUI framework and board display
* Member 2: Implement input handling and basic two-player mode

**Team B:**

* Member 3: Implement basic Minimax algorithm and game tree
* Member 4: Implement dataset processing and basic ML algorithm

**Deliverable:** Working individual components (not yet integrated)

**Week 3 (Oct 29-Nov 4): Component Completion**

**Team A:**

* Member 1: Complete GUI features, enhance display systems
* Member 2: Complete two-player mode, implement game flow management

**Team B:**

* Member 3: Complete Minimax optimization, implement one-player mode
* Member 4: Complete ML training, implement imperfect AI

**Deliverable:** Fully functional individual components ready for integration

**Phase 2: Integration & Finalization (3 weeks)**

**Week 4 (Nov 5-11): Integration & Initial Testing**

* **Member 1:** Begin comprehensive testing of all components
* **Member 2:** Integrate all components into main system
* **Member 3:** Begin performance analysis and algorithm comparison
* **Member 4:** Start report structure and documentation framework

**Deliverable:** Integrated working system with initial performance metrics

**Week 5 (Nov 12-18): Testing & Optimization**

* **Member 1:** Complete cross-platform and user testing
* **Member 2:** Optimize system performance and handle integration issues
* **Member 3:** Complete performance analysis and generate comparison reports
* **Member 4:** Complete major report sections and begin presentation preparation

**Deliverable:** Fully tested and optimized system with comprehensive documentation

**Week 6 (Nov 19-25): Final Documentation & Submission**

* **All Members:** Collaborate on final report sections and presentation
* **Member 4:** Compile final submission package and coordinate final review
* **All Members:** Create presentation video and demo materials

**Deliverable:** Complete submission package ready for deadline

**Phase Transition Meeting (November 4-5)**

**Critical Handover Session:**

* **Team A** demonstrates completed GUI and two-player components
* **Team B** demonstrates completed AI and ML components
* All members review code structure and interfaces
* Plan integration strategy and identify potential issues
* Reassign any incomplete tasks
* Set Phase 2 priorities and deadlines

**Advantages of Two-Phase Approach**

1. **Focused Development:** Team members can deeply focus on their core expertise areas
2. **Parallel Progress:** GUI and AI development proceed simultaneously without dependencies
3. **Natural Skill Development:** Members develop specialized skills before broadening to testing/integration
4. **Risk Mitigation:** Major components completed early, leaving time for thorough testing
5. **Clear Transition Point:** Natural break allows for progress assessment and strategy adjustment

**Success Metrics by Phase End**

**Phase 1 Completion Criteria:**

* Working GUI with complete two-player functionality
* Functional Minimax AI with one-player mode
* Trained ML algorithm with imperfect AI capability
* All components individually tested and documented

**Phase 2 Completion Criteria:**

* Fully integrated system tested across multiple platforms
* Comprehensive performance analysis completed
* Complete documentation and presentation materials
* Submission package ready for deadline

This revised structure maximizes parallel development efficiency while ensuring thorough integration and testing in the final phase.