General Critique

Overall

- **UML diagram is incomplete. The classes are mostly there but you are missing many dependency/relationship arrows.**
    - To improve the readability and enhance the organization of your organization it is recommend to fill in all the dependency arrows
- **Missing Database design in networking**
    - Your design does not specify the type of database or how it will store data, it is recommended to design a schema on paper to understand how to build database architecture to ensure optimized data structures.
- **Improvements on use case diagram**
    - Use case diagrams need system boundaries.
    - Standardize formatting throughout entire document
- **The game "Whist" is not an option on the games list.**

GUI

Strengths:

- The mid-fi planning is thorough with intuitive, user-friendly mockups.
- GUI components are clearly defined, with consistent use case descriptions and attention to color and layout.

Areas for Improvement:

- **Unique Design & Integration:**
  – Consider adding unique design elements beyond basic mockups.
  – Develop clear documentation (e.g., a diagram) showing how the GUI will connect with underlying game logic and integrate with other teams' code.
- **Quit Game Functionality:**
  – Rather than having two distinct quit game use cases, merge them into a single, flexible UI path.
  – Ensure quitting does not force a hard quit; incorporate notifications (e.g., for forfeits or temporary bans in competitive matches).

- **Navigation & Layout:**
  – Add a navigation flow diagram or detailed description of screen transitions.
  – Align vertical boxes (e.g., centering under “Welcome Back”) for a cleaner layout.
- **Additional Information Screens:**
  – Include an instruction scene available before and during gameplay.

Game Logic

Strengths:

- Acknowledges that each game (e.g., Checkers, Connect 4, Tic Tac Toe) requires its own setup and specialized logic.
- Emphasizes the potential benefits of using polymorphism for piece types.

Areas for Improvement:

- **UML & Diagram Clarity:**
  – The UML diagram is incomplete; update it to include all dependency/relationship arrows.
  – Ensure that game-specific classes (e.g., Connect 4) are correctly depicted and do not incorrectly extend from unrelated classes (e.g., a card class extending GamePiece).
- **Generalization & Naming:**
  – Avoid violations of the generalization principle by correctly aligning class hierarchies (e.g., correct the “GamePeice” typo to “GamePiece”).
  – Revisit the enum for ColourType—ensure it reflects the proper constraints (e.g., only two colours/types for checkers).
- **Use Case Detail & Move Validation:**
  – Include standard game logic use cases (e.g., updating game state and defining win conditions) in the use case documentation.
  – Detail the initial setup per game (e.g., board dimensions for Connect 4, grid for Tic Tac Toe) rather than a vague “initial setup.”
  – Add move validation use cases to ensure rules aren’t overlooked during coding.
- **Gameplay Mechanics & Deadlines:**
  – Replace overcomplicated processes (e.g., drawing two checkers and a 10-second selection timer) with simpler alternatives like alternating turns or assigning starting roles based on join order.
  – Incorporate personal deadlines in the planning document to allow time for compatibility and communication between teams.

- **Attribute Visibility & Valid Moves:**
  – Standardize attribute visibility by making fields private and providing getters/setters for encapsulation.
  – Consider maintaining a list of valid moves at each game stage to reduce on-the-fly validation complexity.

Networking

Strengths:

- The use case diagram covers core functions like leveling, matchmaking, and basic communication.

Areas for Improvement:

- **Integration & Timeline:**
  – Clarify the final integration date (e.g., determine if it should be April 11th rather than March 11th).
- **Security & Database Design:**
  – Implement network security measures to protect against unauthorized access or data breaches.
  – Develop and document a detailed database schema to ensure the design can store data efficiently.
- **Use Case Completeness:**
  – Expand networking use cases to address scenarios such as server failures, network timeouts, and game synchronization (e.g., handling mid-turn disconnections).
  – Clearly depict how the server will manage the flow of messages between users.

- **Account & Communication Enhancements:**
  – Consider adding an account deletion mechanism.
  – Enhance the use case descriptions to include gameplay communication details.

Authentication

Strengths:

- Basic authentication and profile use cases are present and include primary user roles.

Areas for Improvement:

- **Expanded Use Cases:**
  – Include missing authentication actions such as account deletion, sending and accepting friend requests, profile searches, and tracking game history.

– Add use cases for updating player stats as part of the gameplay experience.

- **Diagram Detail & Accuracy:**
  – Enhance the use case diagrams with more descriptive names (e.g., “View Profile” instead of just “Profile”) and include extending use cases for updating profile details.
  – Correct diagram inaccuracies such as misdirected extend relation arrows, ensuring the Logout use case is linked to Settings appropriately.
  – Represent the Database as an actor (if applicable) rather than a use case, and include all secondary actors (e.g., system administrators, existing vs. new players).
- **Consistency in Terminology:**
  – define each use case's expected date of availability

Leaderboard & Matchmaking

Strengths:

- There is a solid foundation with creative ideas for both the leaderboard and matchmaking systems.

Areas for Improvement:

- **Leaderboard:**
  – The leaderboard class is detached in the class diagram; integrate it by adding necessary fields (e.g., player details) and clearly linking it to other project components.
  – Establish a clear creation timeline and development plan.
  – Specify functionality details such as what metrics are shown, how players interact with it, and the default sorting options (e.g., by level, game points).
- **Matchmaking:**
  – Expand use cases to provide a step-by-step description of the matchmaking process, including handling of mid-game exits and friend-based matches.
  – Eliminate redundant use cases (e.g., merge “Matchmake Highly Experienced Player” with “Match Players based on Experience Level”) for clarity.
  – Update class diagrams to show dependencies for starting a game, linking the Player class and other related components.
  – Define the data structures or algorithms to be used in matchmaking to address open issues and streamline the process.

The Lima Team's project demonstrates strong foundational work in multiple areas, including UML diagrams, GUI mockups, and game logic design. The team has successfully outlined core features such as authentication, networking, and matchmaking. However, several critical areas require refinement, including missing UML dependencies, inconsistent use case formatting, and the absence of a database schema. GUI navigation flow needs clearer planning, and integration timelines across teams should be more structured.

Key improvements include refining authentication use cases, adding detailed matchmaking algorithms, and improving security considerations. The networking and database design require more depth, particularly in failure handling and account management. Additionally, refining the game logic structure and standardizing attribute visibility will enhance maintainability. Addressing these areas will improve system coherence and ensure seamless integration across components.

Planning Documents Review

Overall the timeline for GUI, Statistics, and Game logic show tasks scheduled and have reasonable deadlines set for completion. However, these documents can be improved by detailing the classes that they wish to complete by given deadlines. Additionally, there is little mention of test files being created in planning documentation. In the networking timeline there is a deadline for integrating all structures on March 11th, it is better to keep this as April 11th to ensure you've completed all prior tasks before fully integrating software.

The team does not outline their risks and dependencies in their planning documents.

Feature Requests

Feature 1: In-Game Tutorial & Interactive Hints

Purpose:

This feature introduces an interactive tutorial and real-time hints for new players. It helps users understand game mechanics, controls, and strategies without needing external guides.

Feature 2: Player Statistics & Performance Tracking

Description

This feature introduces detailed player statistics, allowing users to track their progress over time. Metrics include win/loss ratio, average game duration, best moves, and a leaderboard ranking system.