Critique Received regarding Accounts/Networking

- <Critique (their comments)>
    - <Possible resolution or comment (my comments)>

Review 1:

\review_1\design\Lima_annotated_class_diagram.png

- `UserAccount` and `PlayerAccount` classes not consistent or conflict with each other
    - We **do** need to unify this and make sure we know how our isolated sections can combine (how accounts are saved, interact with the games, etc.)
- Missing relation to Game in `Player` class
    - We should show how this combines or is implemented within the larger picture, as there are no connections shown
- Errors in `UserAccount` method syntax
    - Will naturally be resolved during coding
- Use `GameType` enum to identify games in player class
    - My intention linking `GameSession` (and others) to `Game` was not intended to imply that there would be another `Game` class here, rather that it 'links' to the Game module. I like the idea of consistently using the `GameType` enum across our platform

\review_1\design\Lima_annotated_use_case_descriptions.pdf

- Add a `SearchForProfile` functionality for Statistics to help with friend requests and profile viewing
    - Seems like a good idea, but low priority since it won't affect core functionality
    - Would need to consider how to search (eg. username, email, etc.)
- Need clearer deadlines for milestones
    - Our meeting this past Monday addressed this
- Viewing past games may cause issues if the date for doing this has been passed
    - Could base this on "the last game played" regardless of time, rather than based on a recent date. Would facilitate when to purge or remove records from the DB as well (when a new record is added, rather than a trigger at a certain date)
- Add servers to secondary actors (in Tracking Game History use case)
    - Sure
- Use Case: Sending a Friend Request is not critical to gameplay or functionality, and could be pushed to a later iteration
    - Agree with this, lets focus on an MVP
- Unclear what the maximum number of friend a player can have is

    - This would be based on space considerations, how much we would possibly want to allocate for each `UserAccount`
- Use case: Forgotten Password Reset; what happens if the user can't access email on their account
    - We should consider this being the case where an account is irretrievable, and a user would need to create a new account (with an email address they have access to) in order to continue playing.
- Use case: Logging out of an account; should the user be auto-logged out when they close the program?
    - This is a good idea, otherwise we'd risk a "hard" logout (such as when the connection fails). Could add as one of the last steps to an `onExit` trigger

\review_1\feature\Lima_FeatureProposal.pdf

- Player Statistics and Performance Tracking; implement more detailed stats (eg. win/loss ratio, average game duration, best moves, leaderboard ranking)
    - Some of these will be easy (win/loss ratio), but saving "best moves" seems unnecessary, and only worth considering at a later iteration (time permitting)

\review_1\LimaLettertoDevTeam.pdf

- Diagrams are incomplete: missing dependency/relationship arrows
    - We should double check all diagrams and fill these in where appropriate. Very few classes will have ZERO connections, so these should stand out as needing work
- Missing database design
    - We are somewhat implementing a database, and we can show the proposed schema, however this was explicitly mentioned in the project description as something we didn't need to worry about, so I find it odd that they would comment on this
- Unclear about how our different modules connect
    - Need to make sure we have appropriate handlers and consistent classes
- Clarify integration plan and timeline
    - In progress…
- Include additional use cases to address server failures, network timeouts, and in-game desynchs
    - Should these be separate use cases, or exceptions to intended use cases?
- Add account deletion mechanism
    - We have this, whoever wrote this letter must have missed it
- Expand use cases to include friend requests and profile search
    - Time permitting these are fine ideas, but neither contribute to the MVP, so should be considered low-priority
- Integrate `Leaderboard` into the rest of the class diagrams
    - As mentioned above, all classes need to connect to the project
- Expand Matchmaking use cases

    - This will happen naturally as we code, so we'll be able to develop a better use case description/diagram after this work has been completed

Review 2

Note: I will attempt to not repeat issues mentioned in Review 1

\review_2\design\Lima - Annotated Use Case Descriptions.pdf

- Use case description: player matchmaking; primary actor should be player, not game platform
    - Seems valid, we can change this
- Need authentication use case diagram
    - Sure

\review_2\design\Lima - Design Corrections and Feedback.pdf

- Connect `Leaderboard` to `UserAccount` class
    - We are in the midst of changing from passing via strings to passing objects, which will help this (and similar) issues
- Game chat connection is unclear (relates to `GameSession`)
    - Will be clarified during coding/implementation, can update the diagrams as required

\review_2\design\Lima - structure_diagrams_annotations.pdf

- Connections missing
    - Our individual modules should connect so we know how everything is meant to fit together
- Missing methods (eg. `+ addPlayer(player: UserAccount)`) (in `GameSession`)
    - Is this not the reference? I don't understand the comment

\review_2\feature\Lima - Feature Proposal.pdf

- Requested Feature: Session Owner Kick Control
    - Good idea, we should try to implement
- Requested Feature: Spectate Mode
    - Interesting idea, but I would rank this as very low priority
    - Might require two `GameSession` modes
- Improvement: filter through leaderboard
    - How much would we want to load from the DB (need to balance detail provided and performance of the platform)
    - Could consider for a later iteration
- Improvement: username moderation system
    - A simple blacklist would be effective for the most blatant cases (eg. "normal" swear words), but an exhaustive blacklist isn't feasible

    - Implementing a “smart” detection algorithm (so “r@cist” is detected as “racist”) seems sketchy, as incorrect detection would cause more frustration from players than occasionally seeing offensive names
    - “Report username” good idea, but should only go to a moderator’s attention (not automated detection). One solution, have the local GUI of a reporting player altered so the offensive name is replaced by “FluffyBunny01”, so that the immediate issue of the offensive name being seen is resolved.
    - Would require moderation
- Improvement: game state auto-recovery on disconnect
    - Implementing an “autosave” snapshot at every turn seems like a good idea