Feature Proposal Lima

<u>Feature 1:</u> Customizable Themes & Accessibility Modes

Description: This feature introduces a customizable theme system to enhance the visual aesthetics and accessibility of the Board Game Platform's user interface. Users will be able to personalize the colors of menus, home screens, and in-game interfaces according to their preferences. Additionally, the system will include preset themes such as Dark Mode, Light Mode, and Accessibility-Friendly (Colorblind) Mode to improve the platform's inclusivity.

Each user's selected theme will be stored and automatically applied upon login, ensuring a consistent and personalized experience. This enhancement aligns with modern UI/UX best practices, catering to both aesthetic customization and functional accessibility.

Expected Impact:

- Personalized interfaces lead to greater user satisfaction and retention. Each person has a sense of identity to their account.
- Users feel a stronger sense of ownership over their gaming environment.
- Colorblind-friendly themes ensure that all players, including those with visual impairments, can comfortably navigate the interface.
- High-contrast and grayscale modes improve readability for users with vision difficulties.
- Ensures that users experience consistent visual quality across various screen sizes and resolutions.
- Reduces eye strain in long gaming sessions by allowing users to choose dark themes or warmer colors.

Suggestions for Implementation/Design:

1. Theme Selection & Preset Options

- Preset Themes:
    - Light Mode: Classic bright UI.
    - Dark Mode: Low-light friendly, reduces strain.
    - Colorblind Mode: Adjusted hues ensuring visibility for all types of color vision deficiencies.
    - High-Contrast Mode: Black/white-heavy contrast for maximum clarity.
- Custom Theme Creation:
    - Users can choose primary, secondary, and accent colors.
    - The GUI dynamically updates previews before saving changes.

2. Profile-Based Theme Persistence

- Users' theme preferences are stored within their profile settings.
- The platform retrieves and applies the saved theme automatically on login.
- Integration with the Settings Menu to allow users to modify themes at any time.

3. Application of Themes Across UI

- A consistent theme will be applied to the following:
    - Dashboard
    - Game Library & Matchmaking Screen
    - Leaderboard
    - Game Screens
    - Chat Windows & Notifications
    - In-Game Menus & Toolbars

- Dynamic CSS variables within JavaFX can enable real-time updates without requiring page reloads.

4. User Experience Enhancements

- Live Preview: Users can preview theme changes in real-time before applying.
- Save & Reset Options: Allow users to revert to default themes if needed.
- Per-Game Themes: Option to allow custom themes per game (e.g., a different theme for Checkers vs. Connect 4). (Optional but could be something worthwhile to maximize customizability)

<u>Feature 2:</u> Session Owner Kick Control

Description:

This feature grants session owners (players who create a game lobby) the ability to remove unwanted players before a Whist match begins. Since Whist requires exactly four players, this ensures that hosts maintain control over their game environment. However, once the match starts, the kick function is disabled to preserve game integrity. If a player disconnects during the match, an AI-controlled bot will replace them to prevent disruptions.

Expected Impact:

- Hosts can curate their gaming experience by removing unwanted players before a match starts.
- Prevents mid-game disruptions, ensuring players cannot be unfairly removed once the game has begun.
- AI replacements handle unexpected disconnects, ensuring a smooth gameplay flow without needing to restart.

Suggestions for Implementation/Design:

- Pre-Game Removal:
    - Add a "Kick Player" button next to each player's name in the lobby screen, visible only to the session owner.
    - Show a confirmation pop-up before kicking a player to prevent accidental removals.
    - Modify the LobbyManager class to handle removals cleanly.
- Rejoin Prevention:
    - The MatchmakingHandler should prevent a kicked player from immediately rejoining the same session.
    - Implement a cooldown timer (e.g., 2 minutes) before they can attempt to rejoin.
- In-Game AI Takeover:
    - Once the match starts, disable the kick function to maintain fairness.
    - If a player disconnects mid-game, the GameSession class will replace them with an AI bot to preserve game flow.
    - Display a notification in the chat: *"[Player X] disconnected. AI has taken over their hand."*

<u>Feature 3:</u> Password Visibility Toggle

Description: Enhance the sign-in page usability by adding a password visibility toggle that allows users to switch between obscured and visible text in the password input field. This feature helps users reduce login errors by verifying their input, especially beneficial for those on mobile devices or with visual impairments.

Expected Impact:

- Improved Usability: Reduces sign-in frustration by allowing users to verify passwords before submitting.
- Enhanced Accessibility: Aids users with low vision, small screens, or assistive technology in entering credentials accurately.
- Minimal Implementation Overhead: A lightweight addition that seamlessly integrates with the existing login flow.

Suggestions for Implementation/Design:

- Placed inside the password field for intuitive access.
- Clicking the icon toggles the field type between "password" (hidden) and "text" (visible).
- Use hover and focus states to improve interactivity and feedback.
- Show a subtle "Password Visible" indicator when toggled.
- Automatically re-hide the password after a short delay (optional for security).

<u>Feature 4:</u> Spectate Mode

Description: The Spectate Mode function will significantly enhance user engagement by allowing users to watch live games in real-time, building a stronger community and facilitating strategic learning. By enabling spectators to observe gameplay without participating, this feature appeals to casual viewers, friends, and competitive analysts who want to analyze moves, predict strategies, or simply enjoy the experience. It also creates social opportunities through a dedicated spectator chat, while ensuring fair play by preventing direct communication with active players.

Expected Impact:

- Spectate Mode allows players to watch friends' matches, fostering engagement and community-driven interactions. It provides opportunities for users to support their friends, discuss strategies, and enjoy games without actively playing.
- By watching high-level play, new and mid-level players can passively develop advanced tactics and decision-making skills, improving their overall proficiency and retention.
- Players can watch friends' match play, offer support, and dissect plays outside of a general conversation. It facilitates friendly competition and encourages additional players to attend multiplayer games.

Suggestions for Implementation/Design:

1. UI & Matchmaking Enhancements
    - Add a "Spectate" tab in the Game Library & Matchmaking screen to display active matches.
    - Provide a list of public games available for spectating, with an option for private matches to be invite-only.
    - Implement real-time board updates with smooth animations to reflect moves instantly.
    - Add a feature to MatchmakingSystem to list active games that can be spectated.
2. Game State & Backend Changes
    - Implement a read-only mode in the GameSession class where spectators receive game state updates but do not interact with gameplay.
    - Modify GameState to allow spectator-specific data updates, ensuring only relevant board information is shared.
    - Add a spectator tracking system to handle viewership count and potential analytics.
3. Permissions & Privacy Controls
    - Allow public games to be spectated by anyone.

    - Enable private game spectating only through player invitations or game settings.
    - Allow players to disable spectators for personal/private games.
4. Chat & Interaction
    - Spectators can chat with each other, but not with players to prevent interference.
    - Modify ChatSystem to have a separate spectator chat from player chat to maintain focus.

Improvements

Improvement 1: Filter Through Leaderboard

Description: Team Lima's existing system is the Leaderboard System that shows player rankings by categories such as highest level attained, most wins, most games played, and best present player, highlighting good statistics to encourage activity; however, the leaderboard is static and just shows current rankings with no background from history or interaction features. This feature adds to the Leaderboard System interactive filters for sorting and showing rankings by game type (Connect 4, Checkers, Whist), time (weekly, monthly, all-time), and specific categories, and includes historical trend tracking to show a player's rank history over time, such as a graph of their "highest level" rank over the past 30 days.

Expected Impact:

- Provides deeper insights into player progress and performance trends. Historical trend tracking helps players see their improvement over time, motivating them to set goals and stay active on the platform.
- Increases player engagement with the leaderboard by making it more interactive and personalized. Interactive filters allow players to explore rankings that matter to them, making the leaderboard a more dynamic and frequently visited feature.

Suggestions for Implementation/Design:

- Extend the Leaderboard class to include filter parameters, allowing dynamic queries to the database to fetch filtered rankings and then update the leaderboard GUI
- Add a LeaderboardHistory class that logs daily snapshots of player rankings for each category and game type, storing them in the database with fields like playerID, date, category, gameType, and rank. Also update the UserAccount class to link to LeaderboardHistory
- Add a button to view a player's history next to their name on the leaderboard. This opens a separate page with a line graph showing their rank progression over the selected time.

Risks & Mitigations:

- Filter complexity may overwhelm casual players. In order to mitigate this there can be a set amount of preset filters on a dropdown menu that players can select from, and include a reset filter in the GUI.
- If the database or Leaderboard class fails to synchronize rankings across different filter combinations, players might see conflicting or outdated information. Implement a centralized ranking update mechanism in the Leaderboard class that precomputes and

caches rankings for all filter combinations at regular intervals, storing them in a dedicated LeaderboardCache table in the database.

Improvement 2: Clear Input Field Button

Description: Introduce a small “clear” button (typically represented by an “×” icon) inside input fields such as in search boxes, searching username to challenge a player, game chat, etc that allows users to quickly erase all text with a single click. This simple enhancement can streamline data entry and reduce frustration.

Expected Impact:

- Users can quickly clear their input instead of manually deleting text, speeding up form interactions.
- This feature saves time and minimizes user frustration, particularly in scenarios with long or error-prone inputs.
- A uniform clear functionality across forms can contribute to a more polished and professional interface.

Suggestions for Implementation/Design:

- Add a small “×” icon inside text input fields that appears when the user starts typing.
- Ensure the icon is clearly visible without cluttering the interface.
- Maintain consistent design and placement across all input fields where this functionality is implemented.

Risks & Mitigations:

Risk: Users might unintentionally click the clear button, erasing text they did not intend to delete.

- Mitigation:Consider implementing a slight delay or confirmation mechanism when the field contains a substantial amount of text. Alternatively, only show the clear button once a certain number of characters are entered.

Improvement 3: Username Moderation System

Description: This improvement prevents players from using inappropriate, offensive, or racist usernames by implementing a username filtering system during account creation and updates. The system will automatically detect and block usernames containing offensive words, slurs, or inappropriate content. Additionally, a report feature will allow players to flag offensive usernames, triggering a review by moderators or an automated system.

Expected Impact:

- Maintains a safe and inclusive gaming environment by reducing exposure to offensive content.
- Prevents players from using racist or offensive names to harass others.
- Enhances platform reputation and encourages wider community participation.

Suggestions for Implementation/Design:

- Implement a UsernameFilter class that checks names against a predefined blacklist of banned words and a smart detection algorithm for variations (e.g., leetspeak like “r@cist” instead of “racist”).
- Modify the UserAccount class to validate usernames at creation and updates.
- Add a "Report Username" button in player profiles and lobbies, triggering a review by moderators or automated detection.
- If a reported username is found inappropriate, force a rename before allowing the player to continue playing.

Risks & Mitigations:

Risk: False positives might prevent legitimate names from being used.

- Mitigation: Allow an appeal process where players can request a manual review if their username is unfairly flagged.

Risk: Players may bypass filters using special characters (e.g., “rac1st”).

- Mitigation: Implement fuzzy matching and pattern recognition to detect obfuscated offensive words.

Risk: Excessive moderation might frustrate players who want creative usernames.

- Mitigation: Provide real-time feedback when entering a username, suggesting acceptable alternatives instead of outright rejection.

<u>Improvement 4:</u> Game State Auto-Recovery on Disconnect

Description: This improvement introduces game state auto-recovery when a player unexpectedly disconnects from an ongoing match. Currently, if a player disconnects, there is no structured way to pause and resume the game seamlessly. This enhancement ensures that when a disconnected player reconnects, they can resume the match exactly where they left off without data loss. The system will leverage existing game state tracking (in Game class attributes like turnHolder, players, score, grid for Connect 4, and board for Checkers) to store and restore game progress automatically.

Expected Impact:

- Minimizes Match Disruptions:
    - Reduces frustration caused by unintended disconnections, ensuring games do not end prematurely.
    - Prevents games from being unfairly voided due to connectivity issues.
- Enhances Multiplayer Experience:
    - Gives players the option to reconnect and resume play, improving overall engagement.
    - Maintains game integrity by ensuring every move remains valid after reconnection.
- Reduces Need for Manual Restarts:
    - Prevents players from having to restart a game from scratch due to accidental disconnections.
    - Ensures fair play by restoring scores, board positions, and ongoing moves.

Suggestions for Implementation/Design:

1. Implement Auto-Save for Active Game State

- Modify the Game class to include game state snapshots that store the following before each turn:
    - Current turnHolder (Player).
    - Game board state (grid for Connect 4, board for Checkers, hands for Whist).
    - Current scores and any temporary variables needed for ongoing calculations.
    - Use serialization or JSON storage to periodically save this data to memory.

2. Detect Player Disconnections & Prompt Auto-Reconnect

- Extend the matchmaking system to detect if a player loses connection unexpectedly.
- If detected, show a “Reconnecting…” screen instead of force-ending the match.

- Allow a 3-minute grace period where the game waits for the player to return before considering it a full disconnect.

3. Restore Game State on Reconnect

- When the disconnected player logs back in, load the last saved game state into memory.
- Use the saved turnHolder variable to resume turns in the correct order.
- Ensure that no additional moves have been executed during the disconnect to prevent desynchronization.

4. Sync with GUI & Network Teams

- Ensure GUI properly reflects a "Waiting for Player to Reconnect" message for the opponent.
- Work with the networking team to maintain a persistent session ID to validate reconnecting users.

Risks & Mitigations:

Risk: Data loss due to unexpected crashes

- Mitigation: Implement periodic auto-save every turn instead of saving only at disconnection.

Risk: Players exploiting disconnects to stall matches

- Mitigation: Add a 3-minute timeout before auto-forfeiting disconnected players.

Risk: Game state desynchronization.

- Mitigation: Ensure state restoration checks for invalid moves before resuming play..