Gin Rummy, With a Twist

Team 17

Qixuan Zhong
Zizeng Li
Qiwen Jiao
Chenrui Hao
Yunze(Figo) Li

Febuary 7, 2025

Content

1 Title of project and team members

Gin Rummy, With a Twist

Team Member	Email	Role
Qixuan Zhong	zhongq7@mcmaster.ca	Developer
Zizeng Li	li124@mcmaster.ca	Backend Developer
Qiwen Jiao	jiaoq2@mcmaster.ca	Frontend Developer
Chenrui Hao	haoc3@mcmaster.ca	Full Stack Developer
Yunze(Figo) Li	li561@mcmaster.ca	Product Manager & Developer

2 Version

Version Number	Description	Date
3	System Verification and Validation Plan	Feb 7, 2025

3 Test Purpose Statement

The purpose of this test plan is to validate the functionality, accuracy, and reliability of the Dozenal Gin Rummy game system. The testing process will evaluate game logic, UI responsiveness, and networking performance to ensure smooth gameplay and correct implementation of the dozenal scoring system. Additionally, the plan will check the user actions against official game rules, detecting any potential illegal moves by individual players and preventing any form of rule-breaking or cheating. The goal is to maintain fair play and ensure a consistent, rule-compliant gaming experience.

4 Component Test Plan

4.1 Unit Test

4.1.1 Game Logic

Test Case: Correct Hand Scoring
Control: Automatic
Initial State: A hand with 12 cards categorized into melds and deadwoods.
Input: A hand with 12 cards
Output:

- Total deadwoods are correctly calculated based on the sum of all the cards not part of melds.
- Melds are correctly identified, larger sums of melds are preferred.

Test Case Derivation: The scoring follows the dozenal Gin Rummy rules.
How test will be performed: Verify the calculated total score matches the expected value of the deadwoods.

— —

Test Case: Correct Detection of Gin or Big Gin
Control: Automatic
Initial State: Player knocks after their turn or at after drawing a card.
Input: A hand of card with no deadwoods
Output:

- The system correctly detects a Gin or Big Gin.
- Player gets the corresponding score.

Test Case Derivation:

- Player achieves a Gin when he has 12 cards in his hand and no deadwoods.
- Player achieves a Gin when he has 13 cards in his hand and no deadwoods.

How test will be performed: Insert the hand of cards into detection of Gin logic, assert the game identifies the condition as Gin or Big Gin.

— —

Test Case: Detection of Illegal Moves
Control: Automatic
Initial State: Player attempts a move against the rule. (e.g. drops the card just draw from the discard pile)
Input: Invalid sequence of moves
Output:

- The game rejects the illegal move.
- An error message pops up to inform the player.

Test Case Derivation: Players must follow the rule of Dozenal Gin Rummy to play the game.
How test will be performed: Simulate a sequence of invalid moves, assert the game will reject all the moves and inform the player.

— —
Test Case: Correct Timing for Allowing Knock
Control: Automatic
Initial State: A player reaches a score that qualifies for clicking the "Knock" button.
Input: The user's score reaches the designated knock threshold
Output:

- The system should allow the player to click "Knock" when the score meets the required threshold.

Test Case Derivation: This test verifies whether the knock logic correctly follows the game rules, ensuring the feature is enabled at the appropriate score.
How test will be performed: The test will involve attempting to knock at various scores, including those equal to or below the threshold.

4.1.2 UI

Test Case: Buttons response as expected
Control: Manual and Automatic
Initial State: Game loaded properly.
Input: Player clicks button if the condition meets(e.g. Start Game, Knock)
Output:

- Button triggers the expected action.

Test Case Derivation: Buttons must be functional when the condition of meets to click the button.
How test will be performed: Manually click buttons if the condition meets, assert the button does the corresponding action as expected.
— —
Test Case: Properly Display of Player Cards and Scores
Control: Automatic
Initial State: Game has started and loaded properly.
Input: Game updates the cards and scores every round based on the game rules
Output:

- Card are displayed correctly for each player.
- Scores are calculated and displayed correctly for each player.

Test Case Derivation: Frontend should reflect the game state accurately.
How test will be performed: Play the game and trigger card and score updates, assert the UI will match the expected game state.
— —
Test Case: Invalid Action Feedback
Control: Automatic
Initial State: It is the player's turn to draw a card.
Input: The player attempts to drop a card
Output:

- A pop-up notification appears, informing the player to draw a card first.

Test Case Derivation: Ensuring the system provides appropriate feedback for invalid user actions.
How test will be performed: When it is the player's turn to draw a card, attempt to drop a card and verify that the system displays the correct feedback message.

— —

Test Case: Card Position Change
Control: Automatic
Initial State: The player already has cards in hand.
Input: The user attempts to change the position of a card within their hand
Output:

- When the user drags a selected card, it should follow the cursor movement.
- Upon release, the card should be placed at the specified new position.

Test Case Derivation: This test verifies the correct implementation of the card movement logic, ensuring smooth drag-and-drop functionality.
How test will be performed: The test will simulate user interaction by repositioning a card and verifying expected behavior.

4.1.3 Match

Test Case: Player Connects to the Same Game Room Correctly
Control: Automatic
Initial State: Player 1 host a game room and Player 2 is ready to join.
Input:Player 2 entered the room code provided by Player 1
Output:

- Both Players connect to the same game room successfully.

Test Case Derivation: The system should handle room connection correctly.
How test will be performed: Simulate Player room creation and connection, assert both players join the same game room and it’s good to play.

— —

Test Case: Player Reconnects to an Ongoing Game
Control: Automatic
Initial State: Player 1 and Player 2 have joined the same game room.
Input: One player accidentally disconnects and rejoins the same room
Output:

- If within the reconnection window, the game resumes.
- If expired, the player is prompted to start a new game.

Test Case Derivation: The system must handle reconnections and session expiration.
How test will be performed: Simulate a disconnect and rejoin, verifying session restoration or new game prompt.

— —

Test Case: Cumulative player scoring
Control: Automatic
Initial State: Player 1 and Player 2 are in the same room.

Input: Players complete multiple rounds
Output:

- System displays match history and cumulative scores after each round.

Test Case Derivation: The system must track and display match records per round.
How test will be performed: Run multiple rounds and verify score updates.

4.1.4 User Authentication

Test Case: User Sign up
Control: Automatic
Initial State: A new user attempts to create an account.
Input: User provides a username and password
Output:

- If the username already exists, notify the user and reject the registration.
- If the information is valid, register the user and store the securely hashed password.

Test Case Derivation: Ensure the authentication system correctly handles new user registrations.
How test will be performed: Conduct test of both positive and negative scenarios, verify the response messages and ensure passwords are stored securely.

— —

Test Case: User Login
Control: Automatic
Initial State: A registered user exists in the system.
Input: User enters their username and password
Output:

- If the credentials match a record, the user logs in successfully.
- If the credentials are incorrect, notify the user of invalid login details.

Test Case Derivation: Validate the authentication system's ability to verify user credentials.
How test will be performed: Conduct test of both positive and negative scenarios, verify the response messages and error handling.

4.1.5 AI Player

Test Case: AI Bot Makes a Valid Move
Control: Automatic
Initial State: The AI bot is in an active game session.
Input: The bot decides whether to draw from the stack or drop zone and makes a move
Output:

- The bot selects the move based on logic defined.
- The game state updates correctly.

Test Case Derivation: Ensure the AI bot makes logical, valid moves.

How test will be performed: Simulate different scenarios and verify the bot's decisions align with game rules.

4.2 Performance Tests and Metrics

4.2.1 Latency

Test Case: Latency
Control: Automatic
Initial State: Two players are connected in the same game room, and the game started correctly.
Input: Simulate player actions in different networking conditions

- 50ms latency
- 100ms latency
- 200ms latency

Output:

- Measure the time taken for player actions to reflect into the game state in different networking conditions.
- Ensure the latency does not exceed 200ms.

Test Case Derivation: Latency exceeding 200ms could be noticeable and disrupt gameplay in a realtime two player card game.
How test will be performed:

- Use a network simulation tool to introduce different latency for the network environment.
- Record the response time and actions performed by players.
- Record the general game experience in different network latency.

4.2.2 Scalability

Test Case: Scalability
Control: Automatic
Initial State: The server is running properly and having 0 game rooms currently.
Input: Simulate a creation of multiple game rooms for active players.

- 5 game rooms
- 10 game rooms
- 20 game rooms

Output:

- Record the server performance.
- Ensure the server remains stable and multiple games are running properly.

Test Case Derivation: The game should support multiple games running simultaneously without crushing and delay in the response for the server.
How test will be performed:

- Simulate multiple users joining game rooms and playing the game at the same time using a load-testing tool.

- Monitor server performance under different loads.
- Ensure the server performs well under different loads without any crushes occurring or significant delays.