

# Unit testing ([implementation](#))

```
Rummy — pipenv shell › zsh — 80x54

[(Rummy) About 40 Chromebooks ~/Rummy % python3 test/test.py
test_canWin (__main__.TestAILogic) ... ok
test_canWin_discardRequired (__main__.TestAILogic) ... ok
test_cannotWin (__main__.TestAILogic) ... ok
test_cannotWin_discardRequired (__main__.TestAILogic) ... ok
test_findLay_discardTrap (__main__.TestAILogic) ... ok
test_findLay_discardTrap_wild (__main__.TestAILogic) ... ok
test_findLay_ignoreWild (__main__.TestAILogic) ... ok
test_findLay_run (__main__.TestAILogic) ... ok
test_findLay_set (__main__.TestAILogic) ... ok
test_findLay_wild_only (__main__.TestAILogic) ... ok
test_findLay_wild_win (__main__.TestAILogic) ... ok
test_findLay_winTrap (__main__.TestAILogic) ... ok
test_findLay_winTrap_wild (__main__.TestAILogic) ... ok
test_findMelDs (__main__.TestAILogic) ... ok
test_findMelDs_allWild (__main__.TestAILogic) ... ok
test_findMelDs_wild (__main__.TestAILogic) ... ok
test_illegal_1Card1Wild (__main__.TestRules) ... ok
test_illegal_2Wilds (__main__.TestRules) ... ok
test_illegal_4SetAndWild (__main__.TestRules) ... ok
test_illegal_dupSet (__main__.TestRules) ... ok
test_illegal_mixed3 (__main__.TestRules) ... ok
test_illegal_overflowRun (__main__.TestRules) ... ok
test_illegal_overflowRunWild (__main__.TestRules) ... ok
test_illegal_setOverLimit (__main__.TestRules) ... ok
test_illegal_setOverLimit4 (__main__.TestRules) ... ok
test_illegal_smallRun (__main__.TestRules) ... ok
test_illegal_smallSet (__main__.TestRules) ... ok
test_legal_3Set2 (__main__.TestRules) ... ok
test_legal_3Wilds (__main__.TestRules) ... ok
test_legal_4Set (__main__.TestRules) ... ok
test_legal_4Set2 (__main__.TestRules) ... ok
test_legal_aceHighRun (__main__.TestRules) ... ok
test_legal_aceLowRun (__main__.TestRules) ... ok
test_legal_limit3Run (__main__.TestRules) ... ok
test_legal_limit4Run (__main__.TestRules) ... ok
test_legal_limitSet3 (__main__.TestRules) ... ok
test_legal_limitSet4 (__main__.TestRules) ... ok
test_legal_maxRun (__main__.TestRules) ... ok
test_legal_maxRunWild (__main__.TestRules) ... ok
test_legal_middleSet (__main__.TestRules) ... ok
test_legal_mixedRun (__main__.TestRules) ... ok
test_legal_mixedRun2 (__main__.TestRules) ... ok
test_legal_normalRun (__main__.TestRules) ... ok
test_legal_normalSet (__main__.TestRules) ... ok
test_legal_run2Wilds (__main__.TestRules) ... ok
test_legal_shuffledSet (__main__.TestRules) ... ok
test_legal_wildRun (__main__.TestRules) ... ok
test_legal_wildSet (__main__.TestRules) ... ok

-----
Ran 48 tests in 0.004s

OK
```

# Use case testing

**Case:** Add AI player

**Main Scenario:**

1. User sends “add AI player” request
2. Server verifies that the game has not started
3. Server verifies that there is room in the lobby
4. Server adds AI player to game
5. Server notifies other players of the new lobby

**Alternative Scenarios:**

- 2a: Server sees that game has started
- 2b: Server ignores request
- 3a: Lobby is full
- 3b: Server ignores request

**Test Cases:** (100% coverage)

(main) Add an AI player to a lobby

- 2a. Try to add an AI player to a lobby after the game has started
- 3a. Try to add an AI player after the lobby is full of players

**Case:** User joins game

**Main Scenario:**

1. User sends “join game” request to server
2. Server verifies that the game has not started
3. Server verifies that there is room in the lobby
4. Server removes player from their current lobby
5. Server adds the player to the new lobby
6. Server notifies other players of the new player

**Alternative Scenarios:**

- 2a: Server sees that the game has started
- 2b: Server ignores the request
- 3a: Lobby is full
- 3b: Server ignores request

**Test Cases:** (100% coverage)

(main) Join a game as normal

- 2a. Send a join request to a lobby that’s already started a game
- 3a. Send a join request to a full lobby

**Case:** User creates meld

**Main Scenario:**

1. User sends “meld” request
2. Server verifies that the game is in play
3. Server verifies that it is the player’s turn
4. Server verifies that the player has drawn
5. Server verifies that the meld consists of 3 or more cards
6. Server verifies that all cards are in the player’s hand
7. Server verifies that the meld is a run or a set
8. Server moves the cards to a new meld
9. Server notifies all players of the state change

**Alternative Scenarios:**

- 2a. The game is not in play
- 2b. Server ignores request
- 3a. It is not the player’s turn
- 3b. Server ignores request
- 4a. The player has not drawn
- 4b. Server ignores request
- 5a. The meld consists of < 3 cards
- 5b. Server ignores request
- 6a. Some cards are not in the player’s hand
- 6b. Server ignores request
- 7a. The meld is not a valid run or set
- 7b. Server ignores request

**Test Cases:** (100% coverage)

(main) Main scenario

- 2a. Try to make a move before the game starts
- 3a. Try to make a meld out of turn
- 4a. Try to make a meld before drawing
- 5a. Try to meld with 2 cards
- 6a. Try to meld with an illegal set of cards (e.g. a 1, a 5, and a 9)
- 7a. Try to meld with cards that are already on the table

**Case:** User empties hand

**Main Scenario:**

1. User uses meld, lay, or discard action to empty hand
2. Server notices that hand has emptied
3. Server tallies hand values of all players
4. Server sends “end” event to all players with hand scores
5. Client adds hand scores to winner (the player that emptied their hand)
6. Server sends “lobby” event to return players to the lobby

**Alternative Scenarios:**

- 2a. Hand has not emptied
- 2b. Play continues as normal (turn moves to the next player if move was a discard)

**Test Cases:** (100% coverage)

(main) Empty a player's hand

- 2a. Take a turn without emptying the player's hand

**Case:** User draws card

**Main Scenario:**

1. User sends "draw" request with a card
2. Server verifies that the game is in play
3. Server verifies that it is the player's turn
4. Server verifies that the player has not drawn
5. Server verifies that the card is either at the top of the deck or the discard pile
6. Server moves card to player hand
7. Server notifies all players of state change

**Alternative Scenarios:**

- 2a. Game is not in play
- 2b. Server ignores request
- 3a. It's not the player's turn
- 3b. Server ignores request
- 4a. The player has drawn already
- 4b. Server ignores request
- 5a. The player is trying to draw a different card
- 5b. Server ignores request

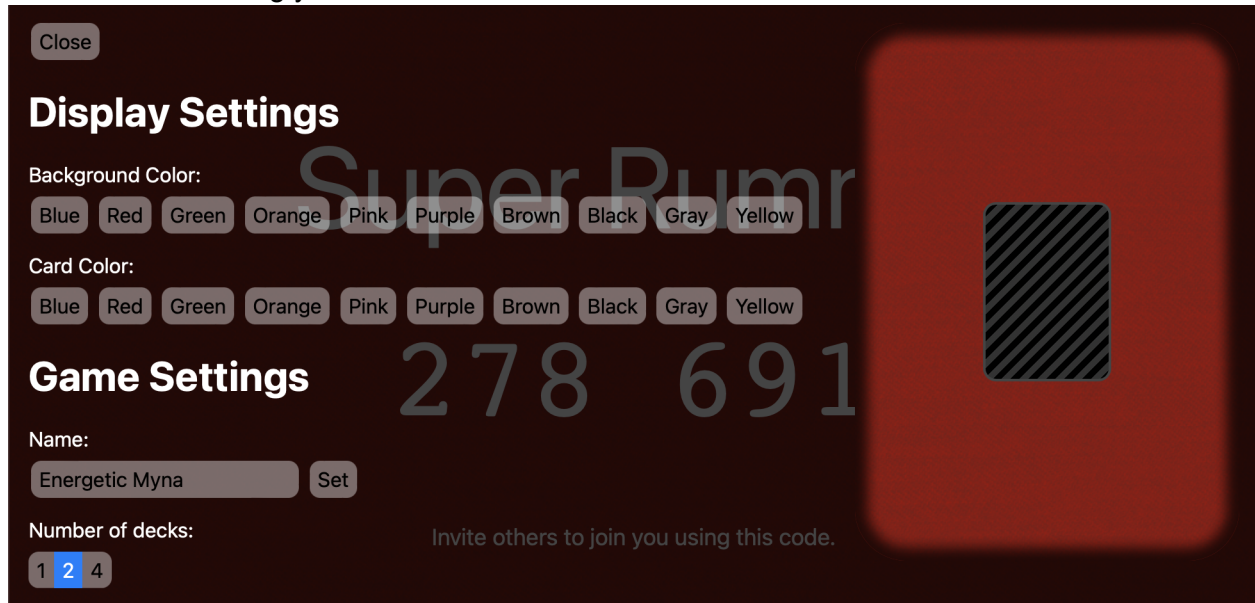
**Test Cases:** (100% coverage)

(main) User makes valid draw move

- 2a. Send a request when the game is still in a lobby
- 3a. Send a request when it's someone else's turn
- 4a. Try to draw twice
- 5a. Try to draw a card from the bottom of the stack

# Acceptance testing

For acceptance testing, we had others play the game, including coworkers and classmates. We noticed some small usability issues that were addressed over time. For example, we noticed when someone tried to customize the colors, that they didn't expect the shade that resulted, and that was a less-than-ideal situation. So, we made a change to make the card and board style visible while choosing your color scheme:



In response, we also adjusted the actual colors to more natural shades.

We also found an issue that the settings weren't accessible during the actual gameplay, and this was frustrating for users. This was also fixed with some updates to the settings interface.