Gin Rummy, With a Twist

Team 17

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

January 24 , 2024

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

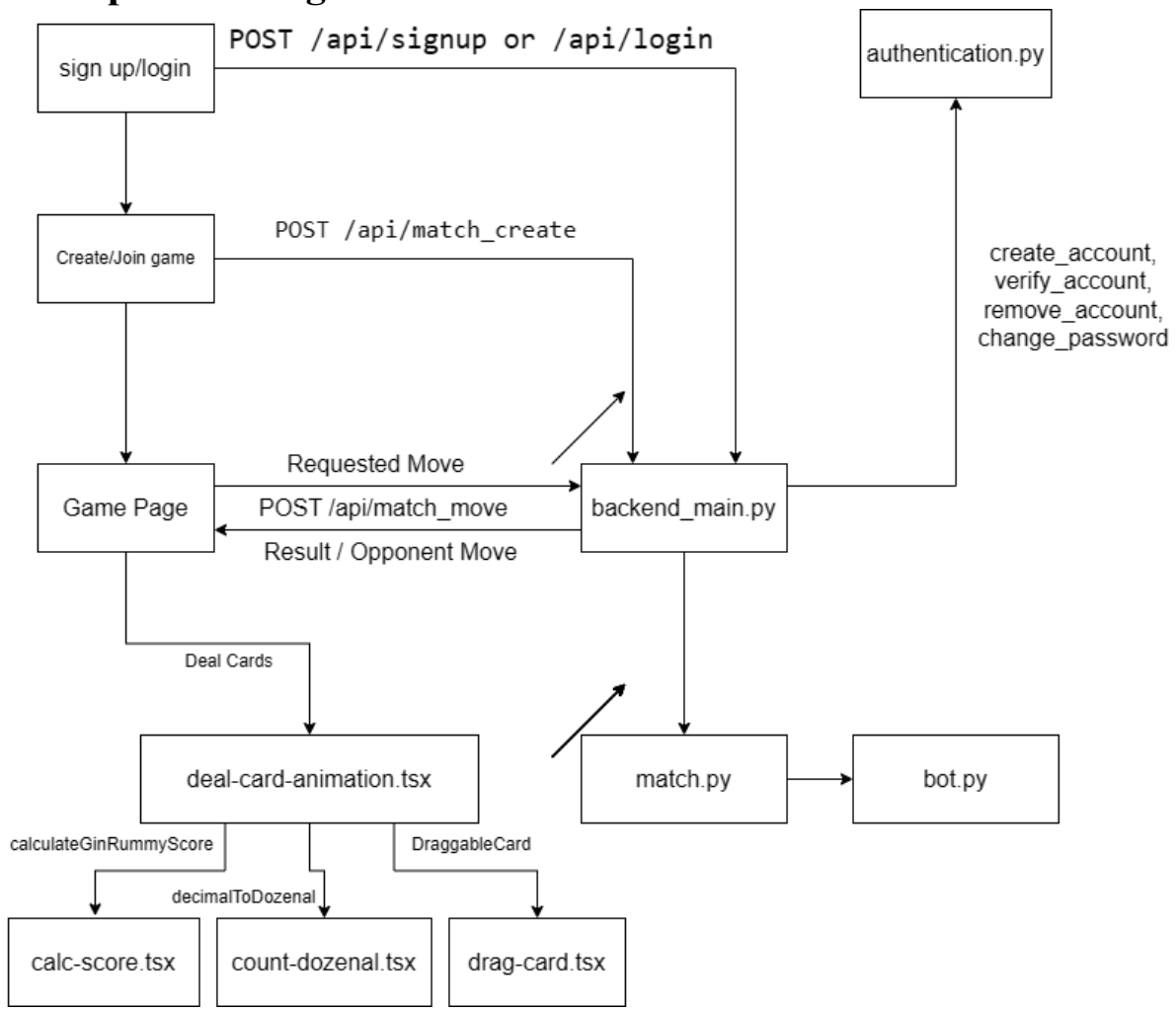
2 Version

Version Number	Description	Date
2	Design Dociment	January 24, 2024

3 Purpose Statement

Gin Rummy has been one of the most popular two-person card games for over a century. This project recreates the game electronically but with a difference. The entire game is played and scored in the dozen number base —“base twelve”— with a dozenal deck of 64 (5 dozen 4) cards. Dozenal has been researched for a few centuries as a number base superior to decimal (“base ten”) mainly because of its abundance of non-trivial factors, 2, 3, 4 and 6, compared to those of decimal, only 2 and 5, with five not being particularly useful.

4 Component Diagram



5 Relationship between Components and Requirements

Components	Requirements
deal-card-animation.tsx	Implement a GUI for the game, preferably a web-based game Highlight the changed cards in dozenal scheme Implement animation for dealing cards, sorting cards and other necessary actions. Ensure the game is playable across web browsers
authentication.py	Implement login feature and user management system
match.py	Implement a lobby feature for inviting two users to the same match

6 Component Description

- Backend

6.1 User Authentication Module

6.1.1 Normal Behavior

- Allows users to sign up, log in, and manage passwords.

- Ensures passwords are stored securely using SHA-256 hashing.

- Prevents duplicate usernames.

6.1.2 API and Structure

- `/api/signup` (POST) → `{username: str, password: str}` → `{result: int, message: str}`

Purpose: Registers a new user by storing a securely hashed password.

- `/api/login` (POST) → `{username: str, password: str}` → `{result: int, message: str}`

Purpose: Authenticates users by verifying username-password matches.

6.1.3 Implementation

- `translated_password(password: str)`: Hashes a password using SHA-256.

- `create_account(username: str, password: str)`: Stores a hashed password if the username is unique.

- `verify_account(username: str, password: str)`: Verifies username-password matches.

- `remove_account(username: str, password: str)`: Deletes user account upon verification.

- `change_password(username: str, old_password: str, new_password: str)`: Changes password securely.

6.1.4 Potential Undesired Behaviors

- Duplicate Accounts: Verify if the username already exists before creation.

- Wrong Credentials: Provide clear error messages for incorrect login attempts.

- Spamming requests: Mitigate repeated requests made using false credentials.

6.2 Match Module

6.2.1 Normal Behavior

- Initializes a card game match.

- Handles card distribution, deck shuffling, and tracking game state.

- Supports player and bot interactions.

6.2.2 API and Structure

- `/api/match_create` (GET) → `{result: int, message: str, match_id: str, cards: list}`

Purpose: Creates a new match, initializes the deck, and assigns initial cards.

- `/api/join` (POST) → `{username: str, match_id: str}` → `{result: int, message: str}`

Purpose: Allows a user to join an existing match.

- `/api/add_bot` (POST) → `{match_id: str}` → `{result: int, message: str}`

Purpose: Adds a bot player to an existing match.

6.2.3 Implementation

- `deck`: Stores the deck of shuffled cards.

- `drop_zone`: Stores the last played card.

- `host_cards`: List of the host player's cards.

- `guest_cards`: List of the guest player's cards.

- `match_id`: Unique identifier for the match.

- `get_initial_cards()`: Retrieves initial card distribution.

- `choose_stack(is_host: int)`: Draws from the stack.

- `choose_drop_zone(is_host: int)`: Picks from the drop zone.

- `drop_card(is_host: int, card_name: str)`: Discards a card.

- `get_latest_operation()`: Retrieves the last move.

6.2.4 Potential Undesired Behaviors

- Game State Corruption: Ensure unique IDs and implement logging.

6.3 AI Bot Player Module

6.3.1 Normal Behavior

- Implements logic for a bot to automatically play the game.

- Decides whether to draw from the stack or drop zone.

- Evaluates the best move to play based on card combinations.

6.3.2 API and Structure

- `/api/match_move` (POST) → `{matchid: str, move: str, host: int, dropped_card_name: str (optional)}` → `{result: int, message: str, card: dict (if applicable)}`

Purpose: Processes a player's or bot's move in the match, updating game state accordingly.

6.3.3 Implementation

- `bot_draw(my_cards: list, drop_zone: dict, stack: list)`: Determines whether to draw from the stack or drop zone.

- `bot_drop(my_cards: list)`: Chooses which card to discard.

- `bot_evaluate(my_cards: list)`: Evaluates the best possible move.

6.3.4 Potential Undesired Behaviors

- Bot Logic Defects: Implement more robust bot decision logic.

- Frontend

6.1 deal-card-animation.tsx

6.1.1 Normal Behavior

- Handles the card dealing animation to start the game, distributing cards to players.

- Manages animations for actions like a user picking a card, dropping a card, or interacting with cards during the game.

- Generate game UI

6.1.2 API and Structure

- `cards.data.ts`: stores all cards

- get_card_from_stack(is_P2: boolean) (POST) : fetch a new card

6.1.3 Implementation

- `dealing`: Currently in dealing animation or not

- `currentPass`: set which player click PASS

- `p1Playing`: record player1's playing status, pick up card or discard.

- `p2Playing`: record player2's playing status, pick up card or discard.

- `player1Cards`: record or player 1's cards on hand.

- `player2Cards`: record or player 2's cards on hand.

- `p1DroppingCard`: record the card that player 1 discard

- `sendingNewCard`: signal for pickup animation

- `remainingCards`: Cards not played yet during this round

- `dropZoneCards`: Cards dropped by players during this round

- `scoreSummary`: score for each player

- resetAll() : clean all record for this round, and start a new round game

- handlePass() : handle the click for PASS in the beginning of the game

- moveCard(fromIndex: number, toIndex: number, wholeCardList: Card[]) : animation for discard

- handleNext() : pick up the next card in remaining cards

- handleDropZone() : pick up the last card in drop zone

- handleDrop(item: { card: Card; index: number }) : discard to dropzone

- handleP1Play() : Connect to backend for another users' play

- handleP1PickAndDrop(newCard: Card) : animation for player 1 to pick and drop a card

- handleKnock() : click KNOCK to end this round

Function Calls:

- Calls `calc-score.tsx` to calculate the score after each round or state change.
- Calls `drag-card.tsx` to allow users to drag to sort their cards.
- Calls `count-dozenal.tsx` to convert decimal counts into dozenal (base-12) numbers.

6.1.4 Potential undesired behaviours

- If calc-score.tsx or count-dozenal.tsx returns unexpected results (e.g., incorrect scores), it could disrupt the game flow.
- Delays in API responses could cause a mismatch between card animations and updated scores.

6.2 calc-score.tsx

6.2.1 Normal Behavior

calculate the score after each round or state change

6.2.2 API and Structure

6.2.3 Implementation

calculateGinRummyScore (CalcScoreCards: CalcScoreCard[]):
PlayerSummary & { Sets: CalcScoreCard[]; Runs: CalcScoreCard[] } :

Calculate all the possible sets and runs according to the player's card. Chose the pattern that has minimum deadwood points

6.2.4 Potential undesired behaviours

If a player wants to choose the pattern that is not the minimum deadwood, they can not combine it by themselves. But the minimum deadwood is always the best strategy to win.

6.3 count-dozenal.tsx

6.3.1 Normal Behavior

convert decimal counts into dozenal (base-12) numbers.

6.3.2 API and Structure

6.3.3 Implementation

decimalToDozenal(decimal: number) : convert decimal counts into dozenal

6.3.4 Potential undesired behaviours

6.4 drag-card.tsx

6.4.1 Normal Behavior

allow users to drag to sort their cards

6.4.2 API and Structure

6.4.3 Implementation

DraggableCard({ card, index, moveCard, p2Playing, wholeCardList }: DraggableCardProps):

drag the card to a new index, and update the whole card list(sort and reorder)

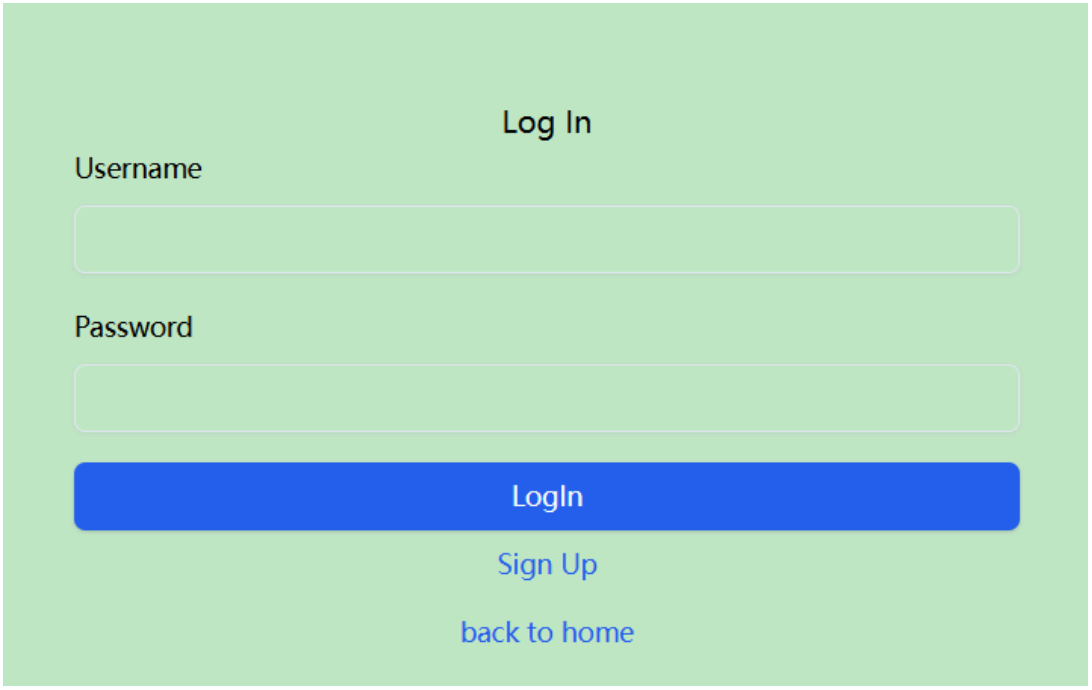
6.4.4 Potential undesired behaviours

if there is a mistake with input card index, the whole list will be changed in a undesired way

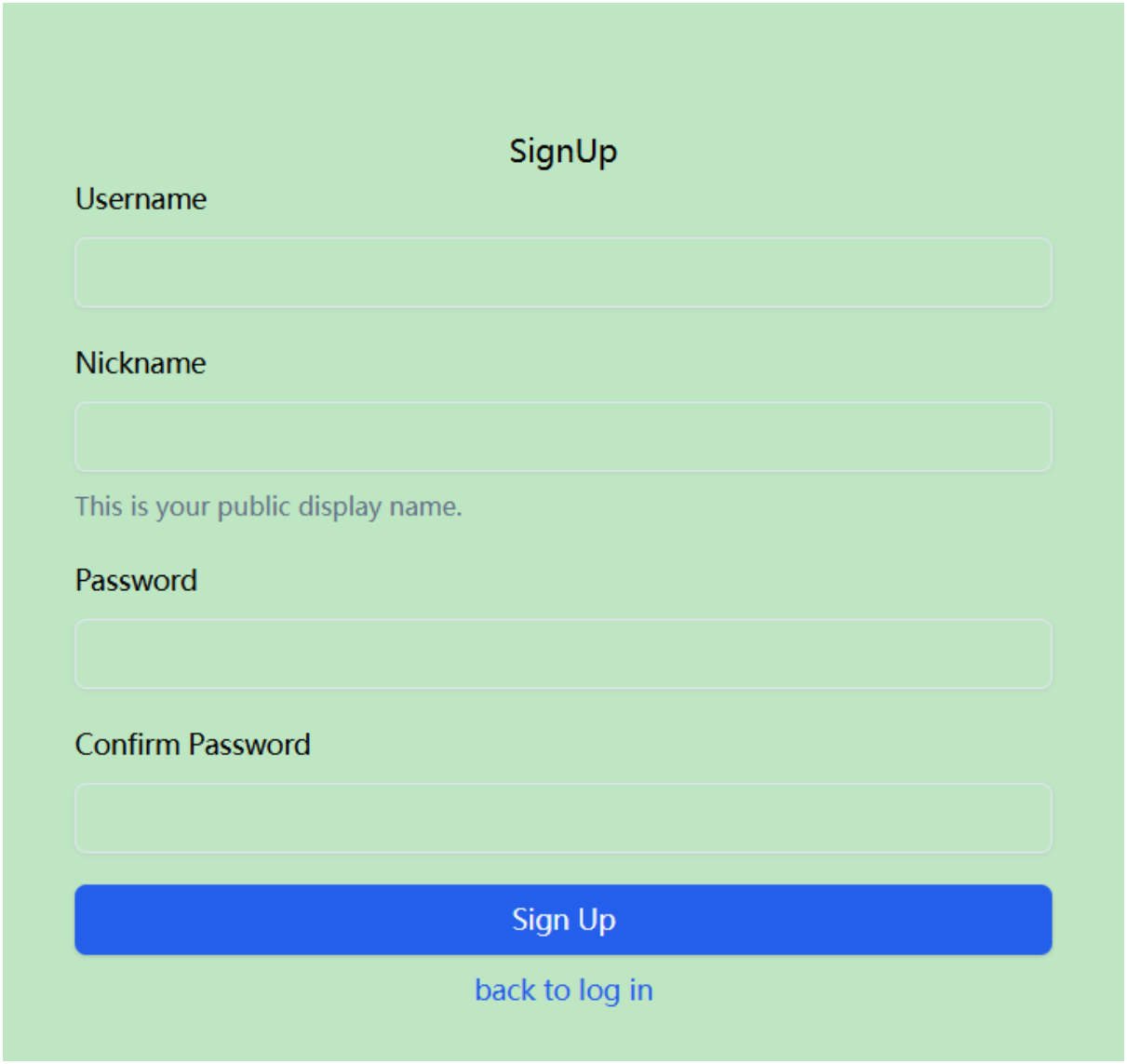
7 User Interface

7.1.1 Home Page

7.1.2 Home Page (click Log In) -> Login interface:

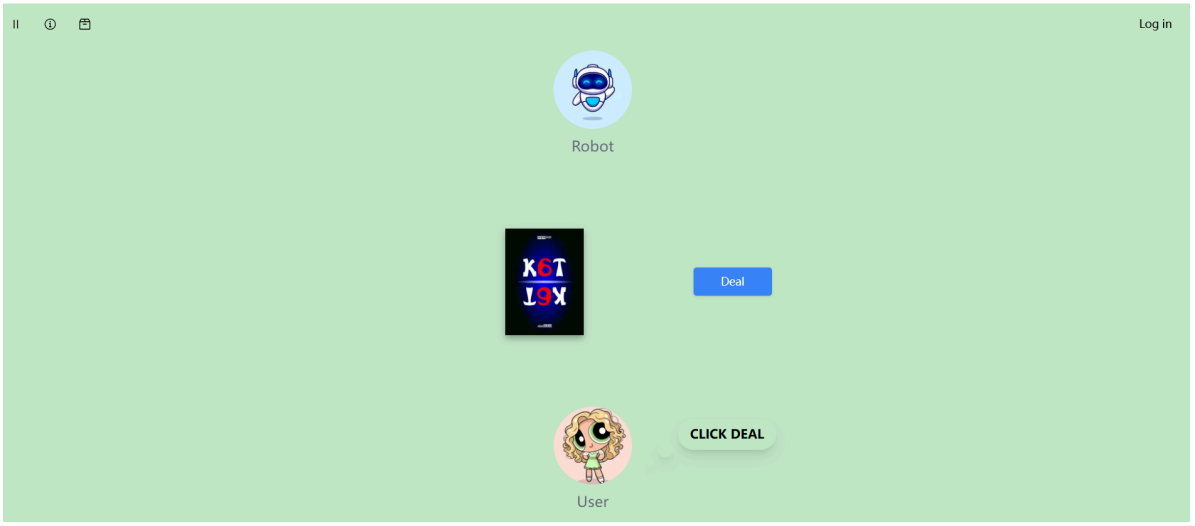


7.1.3 Home Page (click Sign Up) -> SignUp interface

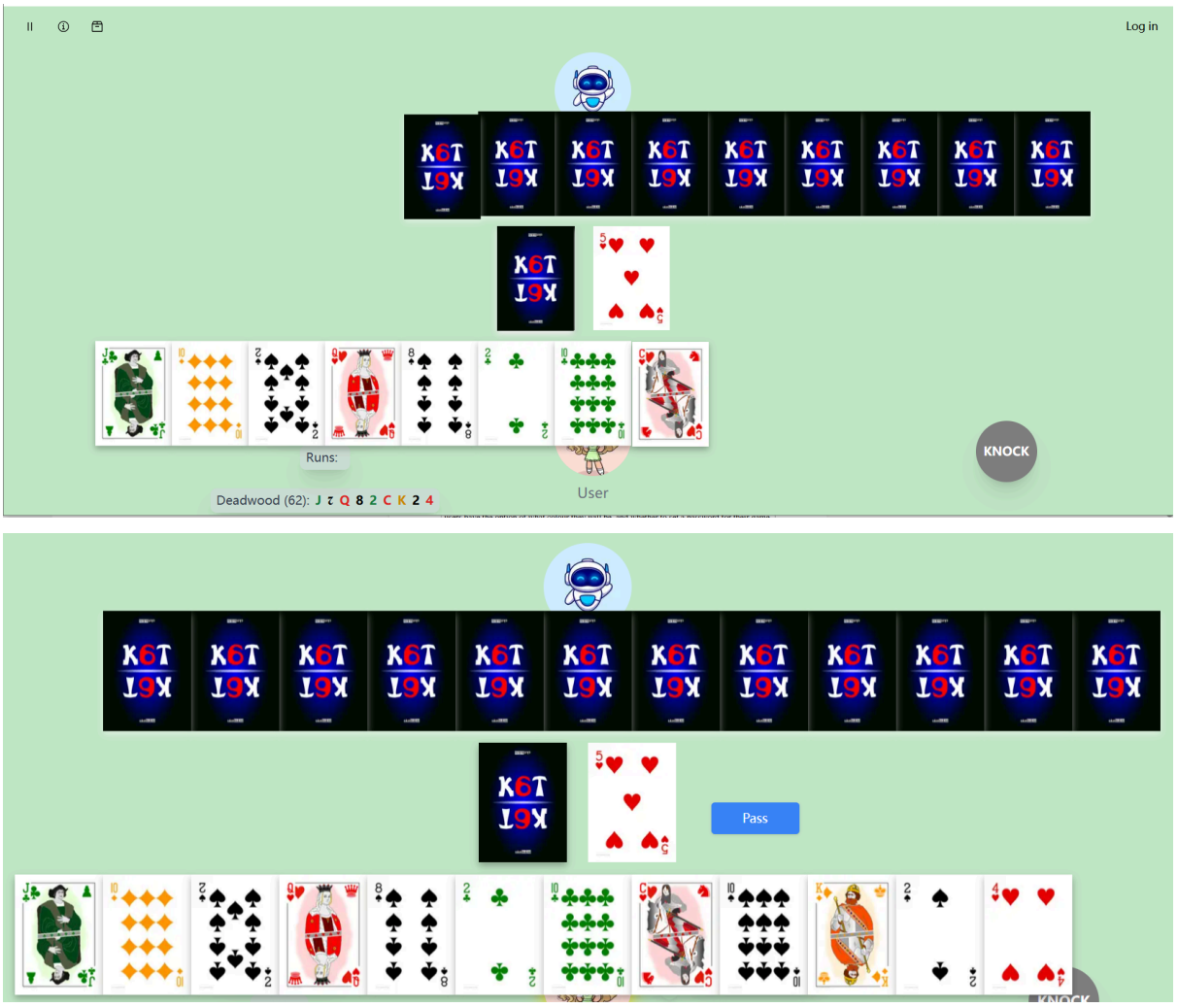


7.1.4 Home Page (Click PLAY) -> Select Mode

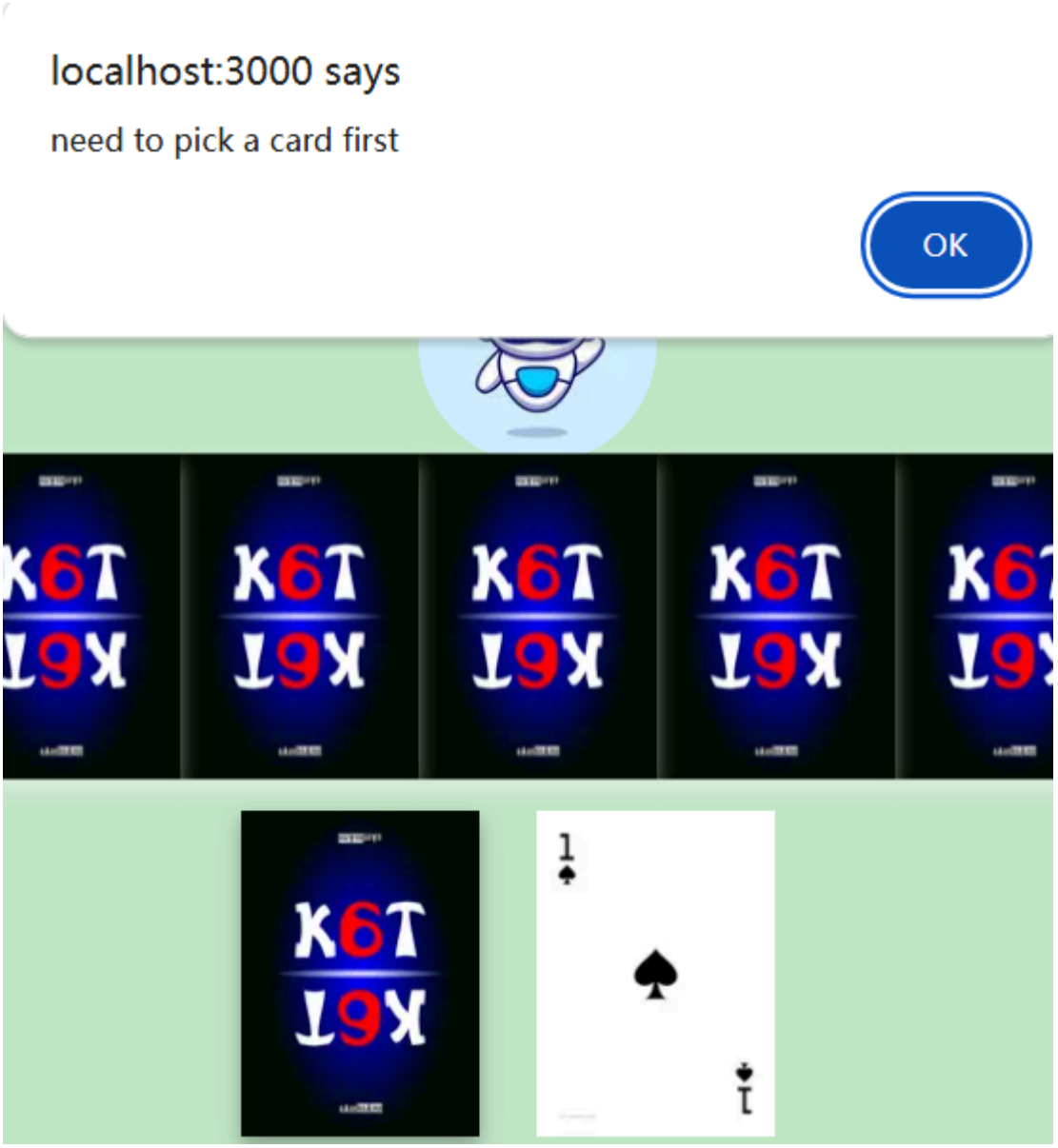
7.2.1 Select Mode (Click any choice) -> Game Room



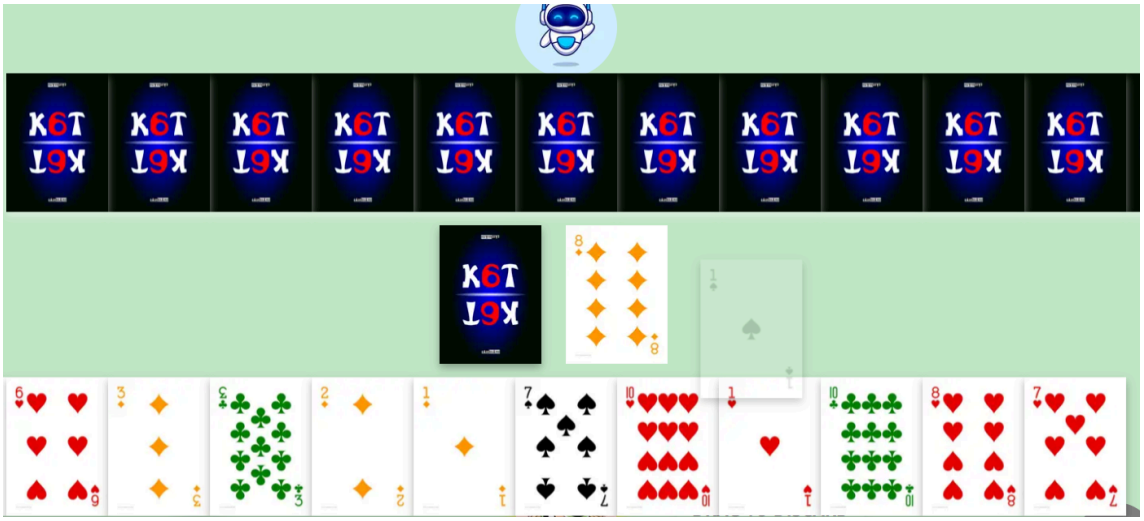
7.2.2 Initialized Game Room (Click Deal) -> Game Start with dealing cards animation



7.3.1 Pop a window alert when user tries to discard a card before drawing a card



7.3.2 Show card move animation when any valid operation is conducted



After: