Team Z-J
20 November 2025
Milestone 3: API Design
Game: Texas Hold' em Poker

I. Game Description

II. Socket.io Events

I. Public Events

A. Game **kataliya**

1. Event: game:hand:started
   Scope: all players in the game room
   Trigger: when a new hand successfully begins after previous hand ended
   Data: {
   "game_id": 123,
   "hand_number": 2,
   "dealer_button_position": 1,
   "pot_total": 30,
   "current_bet": 20,
   "current_player_id":10,
   "phase": "pre-flop"
   }

2. Event: game:hand:dealt
   Scope: specific player
   Trigger: when hole cards are dealt to players at start of new hand
   Data: {
   "game_id": 123,
   "player_id": 10,
   "hole_cards": [
   { "rank": "K", "suit": "hearts" },
   { "rank": "Q", "suit": "diamonds" }
   ]
   }
3. Event: game:state:update
   Scope: All players in the game room
   Trigger: when new hand starts and game state is refreshed
   Data: {
   "game_id": 123,
   "phase": "pre-flop",
   "pot_total": 30,
   "current_bet": 20,

"current_player_id": 10
}

B. Action Taken (all players) (**Junhui)**
    1. Scope: All players in the game
        Trigger: Whenever a player takes an action in the game. (eg. fold / check / call / bet / raise / all-in)
        Data: {
        "game_id": 123,
        "hand_number": 2,
        "round": "pre-flop",
        "User_id": 1,
        "seat": 10,
        "action": "raise", “check”, “fold”...
        "amount": 20,
        "pot_total": 100,
        "next_User_id": 2,
        }

C. Phase updated **Junhui**
    1. Scope: All players in the game
        Trigger: when the game advances to a new phase (e.g., after betting round complete or when everyone is all-in).
        Data
        {
        "game_id": 123,
        "hand_number": 2,
        "pot_total": 150,
        "from_phase": "pre-flop",
        "to_phase": "flop",
        }

D. Betting Round Complete (all players) conditional (**daniel**)
    1. Event Name: game:betting-round:complete
    2. Scope: All players in the game room
    3. Trigger: When a betting round for the current phase finishes, when all remaining players are all in, or when there is only one active player left (everyone folds)
    4. Data: {
        “game_id”: 123,
        “hand_number”: 2,
        “phase”: “flop”,
        “betting_round”: “flop”,
        “pot_total”: 350
        “current_bet”: 100,
        “remaining_active_players”: 4,
        “all_active_players_all_in”: false,

"next_phase": "turn",
"reason": "all_players_matched_bet"

E. Player eliminated (all players) *conditional* ***yuvraj***
   1. Event Name: game:player:eliminated
   2. Scope: All players in the game room
   3. Trigger:When a player's chip count reaches 0 and they are marked as eliminated during or after a hand (e.g., after losing a pot, or between hands)
   4. Data:
      {
      "game_id": 123,
      "player_id": 15,
      "player_name": "John",
      "elimination_reason": "no_chips",
      "players_remaining": 4
      }

F. Game ended (all players) **yuvraj**
   1. Event Name: game:ended
   2. Scope: All players in the game room
   3. Trigger:When the poker game concludes — either because: Only one player has chips remaining, the game reaches a final hand or the table is closed by game owner
   4. Data:
      {
      "game_id": 123,
      "winner_id": 10,
      "winner_name": "Alice",
      "final_standings": [...],
      "pot_awarded": 1200
      }

II. Private event (Acting Player)

A. Chips Updated **yuvraj**
   1. Event Name: game:chips:updated
   2. Scope: Acting player only
   3. Trigger:Whenever a player's **chip count changes**, such as: After betting, After calling, After posting blinds, After winning a pot, and After an all-in adjustment
   4. Data:
      {
      "game_id": 123,
      "player_id": 10,
      "new_chip_count": 750,
      "change": -250,

"action": "bet"
}

B. Hand Deal **yuvraj**
    1. Event Name: game:hand:dealt
    2. Scope: Individual player only, each player receives this event separately.
    3. Trigger:When hole cards are dealt at the start of a new hand.
    4. Dat:
    {
    "game_id": 123,
    "player_id": 10,
    "hole_cards": [
    { "rank": "K", "suit": "hearts" },
    { "rank": "Q", "suit": "diamonds"}
    }

III. API endpoints:

Lobby Managements

A. Create Games(**Junhui**)

HTTP Method & Route	POST /api/lobby/room/
Purpose	Players are allowed to create a room and display it in the public lobby
Authorization	- User must be authenticated - User must be logged in - Game room id must be added
Request Body	{}
Validation Checks	- Game room exists - Game room id is not repeated - Each user can only create one room - Each user can only be in one room (No duplicated user_id) - Max user <= 10

State Updates (What changes in the database?)	- Create a new room record in the database with status waiting. - Assign the user as the owner. - Add the room to the public lobby index - Initialize a Socket.io room channel (room:<roomId>).
Success Response	- 201 Created
Error Cases (What can go wrong)	- 401 Unauthorized - Not logged in - 403 Forbidden - Already created a room - 404 Not Found - Lobby section not found - 400 Bad Request - Validation failed - 409 Conflict - User already creating a room
Socket.io Events (Which Events are triggered?)	1. lobby.room_created 2. room.owner_joined

B. Join Game (**Junhui**)

HTTP Method & Route	POST/Lobby/room/join
Purpose	Allow users to join a room from the lobby
Authorization	- User must be authenticated - User must be logged in
Request Body	{ "observer": false, "buyIn": 100, "seat": null, }
Validation Checks	- Game room id is not repeated - Each user can only be in one room - User id must not be repeated - Max user <= 10

State Updates (What changes in the database?)	- User_id added in the room table - Play_number added to the max_player(max 10)
Success Response	- 200 Accepted
Error Cases (What can go wrong)	- 401 Unauthorized - Not logged in - 403 Forbidden - User already in a room - 404 Not Found - Room section not found - 400 Bad Request - Validation failed - 409 Conflict - Seat_id repeated (2 players join at the same time)
Socket.io Events (Which Events are triggered?)	- Room.user joined - Room.state updated - Lobby.room_updated (play_num incremented)

C. Start Game (**Daniel**)

HTTP Method & Route	Ex. POST /api/games/:game_id/start
Purpose	This endpoint transitions a table from the lobby to an active Texas Hold'em game. The endpoint shuffles the deck, assigns dealer/blinds, posts blinds, as well as deals two cards to each player.
Authorization	- User must be authenticated - User must be a player within the game - User must be allowed to start the game - Game must currently be in "waiting" or in "lobby" state
Request Body	{}
Validation Checks	- The game exists - The user is in the game - The game is in lobby state - The user can start the game - Is there more than 2 players to start the game - Game is not in progress

State Updates (What changes in the database?)	- Create / reset current hand record - Assign dealer and blinds - Post blinds and updates chips - Shuffle deck and deal cards - Update game status - Mark players who are currently in turn or who is in-hand
Success Response	- 202 Accepted
Error Cases (What can go wrong)	a. 401 Unauthorized - User is not logged in b. 403 Forbidden - User is not a player within the game c. 404 Not Found - Game does not exist d. 400 Bad Request - Not enough players to start game.
Socket.io Events (Which Events are triggered?)	1. game:started - Scope: All players in the lobby - Trigger: After validation passes and the hand + blinds are set up.

D. Get Game Status (**Daniel**)

HTTP Method & Route	Ex. POST /api/games/:game_id/status
Purpose	This endpoint returns the current state of a specific game for the requesting player including: - Public info everyone can see such as turn or who is dealer and blind - Private info for the player such as their own cards
Authorization	- User must be authenticated - User must be a player within the game
Request Body	None
Validation Checks	- The game exists - The user is in the game

State Updates (What changes in the database?)	No changes made
Success Response	- 200 OK
Error Cases (What can go wrong)	e. 401 Unauthorized - User is not logged in f. 403 Forbidden - User is not a player within the game g. 404 Not Found - Game does not exist
Socket.io Events (Which Events are triggered?)	Realtime updates handled by other events - game:state:update - game:action:bet - game:betting-round:complete

Game Management

E. Action (**Daniel**)

1. Fold, raise, check, call

HTTP Method & Route	Ex. POST /api/games/:game_id/actions
Purpose	This endpoint handles a single player action within the game such as (fold, check, call, bet/raise) while the server validates if the move is legal, updates chips and pot, advances the turn, and marks the betting round once everyone has acted their turn.
Authorization	- User must be authenticated - User must be a player within the game (game participants) - User must be the current player (player id) - User must not have already folded or all-in for the hand - Game must be within a players turn of play
Request Body	- Action_type: one of either “fold”, “check”, “call”, “bet”, and “raise” - amount: number of chips the player wants to put in. Required for bet and raise but ignored for fold, check, and call

Validation Checks	- The game exists - Game is in a betting or turn phase - User is authenticated with the session - User is a participant of the game - User is the current player in turn - Player hasn't folded or all-in'd Action Specific Checks: a. Common values - Current bet - Player bet - Player chips b. fold - Always allowed for active player c. check - Allowed if curren_bet==player_bet d. call - current_bet>player_bet - player_chip>=(current_bet-playerbet) (or if player goes all in) e. bet - current_bet=0 - amount>0 - amount>=big_blind - player_chips>=amount f. raise - current_bet>0 - amount>0 - amount>=big_blind - player_chips>=amount Checks for Amount: - If amount>=player_chips, treat as all-in If everyone folds except for 1, hand ends, show result then start next game
State Updates (What changes in the database?)	For all successful actions: - Insert a record into actions table for logging: game_id, user_id, action_type, amount 1. Fold

	- Update game_participants SET has_folded=true WHERE user_id=current_player_id AND game_id=:game_id; - If after this, only one non-folded player remains, mark game as ready ready for showdown / endhand - Set current_player_id to next active player (if hand isn't over) 2. Check - No chip or pot changes - Mark that this player has acted this round - Move current_player_id to next player 3. Call - call_amount=(current_bet)-(player_bet) - If player_chips<=call_amount, set call_amount=player_chips and is_all_in=true - UPDATE game_participants SET chips=chips - call_amount, bet_this_round=bet_this_round+call_amount WHERE user_id=current_player_id AND game_id=:game_id; - UPDATE game SET pot_total=pot_total+call_amountWHERE id=:game_id; - Mark has_acted_this_round=true - Move current_player_id to next active player 4. Bet - This is the First bet of the betting round - UPDATE game_participants SET chips=chips=amount, bet_this_round=bet_this_round+amount WHERE user_id=current_player_id AND game_id=:game_id; - UPDATE game SET pot_total=pot_total+amount, current_bet=player_bet+amount,

	last_raise_amount=amount WHERE id=:game_id; - Reset has_acted_this_round=false for all other active players so they have to respond to this bet - Move current_player_id to next active player 5. Raise - Similar to bet, but there is already a current_bet - Reset has_acted_this_round=false for all other active players so they have to respond to this bet - Move current_player_id to next active player 6. End of round book keeping - Server checks for: a. Either chips==0 (all-in) or b. bet_this_round==current_bet and has_acted_this_round==true - If that condition is true, mark the betting round as complete
Success Response	- 202 Accepted
Error Cases (What can go wrong)	400 Bad Request - Action_type mission or invalid. - Amount missing/invalid for bet/raise - Amount is zero or negative when it shouldn't be 401 Unauthorized - User is not logged in 403 Forbidden - User is not a player within the game 404 Not Found - Game does not exist 409 Conflict - Game is not in a betting state

Socket.io Events (Which Events are triggered?)	1. game:action:performed → all players in this game room
	{
	"game_id": 123,
	"player_id": 10,
	"action_type": "raise",
	"amount": 200,
	"current_bet": 400,
	"pot_total": 1200,
	"next_player_id": 11
	}
	
	2. game:state:update → all players in this game room
	- Updated public state: pot, who's turn, which players are folded/all-in, etc.
	3. game:hand:update → acting player only
	- Private info like their new chip count and maybe an all-in flag

F. Next Phase (**Kataliya**)

1. Flop, turn, river

HTTP Method & Route	Ex. POST /api/games/:game_id/deal
Purpose	This endpoint deals the next set of community cards after a betting round completes. It burns a card, deals the appropriate community cards, resets betting state for the new round, and determines the next active player to act.
Authorization	- User must be authenticated - User must be a player within the game (game participants) - Can be triggered by ANY player or automatically by server - Betting round must be complete (all active players have acted and matched current bet)
Request Body	- none
Validation Checks	- The game exists - User is authenticated with the session - User is a participant of the game - Game state is 'active'

	- Betting round is complete - Or all players except one have folded Phase Specific Checks: g. pre-flop - Ready to deal 3 cards h. flop - Ready to deal 1 card i. turn - Ready to deal 1 card j. river - Betting complete - Trigger a showdown
State Updates (What changes in the database?)	For all successful actions: - Log the phase advancement action in actions table with game_id, action_type='deal_phase,' and system triggered flag 1. Pre-Flop (deal 3 cards) - Burn top card, and mark as not in play - Deal 3 cards (flop) and mark them as visible to all players at position 0, 1, and 2 - Update the game's phase to 'flop' - Reset the current betting amount to 0 - Reset all active players 'bet_this_round' to 0 and 'has_acted_this_round' to false - Set 'current_player_id' to the first active player after dealer who hasn't folded or gone all in 2. Flop - Burn top card, and mark as not in play - Deal 1 card at position 3 - Update the game's phase to 'turn' - Reset current betting amount to 0 Reset all active players' betting state 'bet_this_round'=0 'has_acted_this_round'=false - Set 'current_player_id' to first active player after dealer button 3. Turn - Burn top card, and mark as not in play

	- Deal 1 card at position 4 - Update game phase to ‘river’ - Reset current betting amount to 0 - Reset all active players better state - Set ‘current_player_id’ to first active 4. River - Show all 5 cards on the table - Update game phase to ‘showdown’ - Set ‘current_player_id’ to NULL - Triggers the END HAND/SHOWDOWN endpoint - Evaluate hands and award pot 5. All Players but one folded - Skip dealing - Update game phase to ‘ended’ - Award entire pot to remaining player - Reset pot to 0
Success Response	- 202 Accepted
Error Cases (What can go wrong)	400 Bad Request - Betting round is not complete 401 Unauthorized - User is not logged in 403 Forbidden - User is not a player within the game 404 Not Found - Game does not exist 409 Conflict - Game is not in a playing state

Socket.io Events (Which Events are triggered?)	1. game:phase:advanced → all players in this game room
	{
	"game_id": 123,
	“phase”: “flop”,
	“community_cards”: [
	{ “rank”: “Q”, “suit”: “hearts” }
	{ “rank”: “J”, “suit”: “diamonds” },
	{ “rank”: “10”, “suit”: “clubs” }
	],
	“pot_total”: 350,
	“current_bet”: 0,
	“current_player_id”: 10,
	“message”: “Flop dealt - new betting round”
	}

	2. game:state:update → all players in this game room
	- New phase (flop/turn/river)
	- Cards now visible
	- Pot total
	- Current bet (reset to 0)
	- Current player whose turn it is
	- All players' betting status reset for new round
	
	3. game:betting:started → all players in this game room
	{
	"game_id": 123,
	"phase": "flop",
	"current_player_id": 10,
	"message": "New betting round started"
	}
	4. game:turn:notification → current player only
	- Notifies the first player to act that its their turn
	- Includes available actions
	- check/bet for new round

C. End Hand (**Kataliya)**

1. Showdown, Pot distribution

HTTP Method & Route	Ex. POST /api/games/:game_id/showdown
Purpose	This endpoint evaluates all active players' poker hands, determines the winner(s), distributes the pot accordingly, and ends the current hand. It handses both showdown scenarios and fold-out scenarios.
Authorization	- User must be authenticated - User must be a player within the game (game participants) - Can be triggered by ANY player or automatically by server - Must be in showdown phase OR only one player remaining (all others folded)
Request Body	- none

Validation Checks	- The game exists - User is authenticated with the session - User is a participant of the game - Game state is ‘showdown’ - OR only 1 active player remains Specific Checks: - If phase is ‘showdown’ at least 2 players haven’t folded - If only 1 player remains, award pot immediately without hand evaluation
State Updates (What changes in the database?)	For all successful actions: - Log the showdown action in the actions table with game_id, action_type=’showdown’, and system triggered flag 1. All but one player folded - Award the entire pot to the remaining active player without revealing cards - Update the winner’s chip count by adding pot_total to their chips - Reset the game’s pot to 0 - Update game phase to ‘ended’ - Set winner_id to the remaining player - Record game end time - No hand evaluation, winner by default 2. Multiple players remain - Evaluate each player’s hand - Determine winner(s) by active hand ranks - Distribute pot - Update game state - Reveal all active players’ hole cards for showdown display 3. Post-hand cleanup - Mark all players’ hole cards as revealed - Update game statistics/history if tracking wins/losses - Check if any players have 0 chips remaining - Prepare game stat for new hand 4. IF game fully ends

	- If it is final hand: - Add the remaining chips back to each players balance in their user table - Update users.balance = balance + player.chips for all participants - Clean up game participant records if game is over
Success Response	- 202 Accepted
Error Cases (What can go wrong)	400 Bad Request - Game is not in showdown phase and multiple players remain - Betting round is not complete - No active players remain 401 Unauthorized - User is not logged in 403 Forbidden - User is not a player within the game 404 Not Found - Game does not exist 409 Conflict - Game is not in a playing state
Socket.io Events (Which Events are triggered?)	1. game:showdown → all players in this game room { "game_id": 123, “showdown_results”: [{ “player_id”: 10, “player_name”: “Alice”, ‘hole_cards”: [{ “rank”: “A”, “suit”: “spades” } { “rank”: “K”, “suit”: “spades” }, “best_hand”: “straight flush”, “hand_cards”: [{ “rank”: “A”, “suit”: “spades” } { “rank”: “K”, “suit”: “spades” }, { “rank”: “Q”, “suit”: “spades” } { “rank”: “J”, “suit”: “spades” } { “rank”: “10”, “suit”: “spades” }], “Is_winner”: true }, { "player_id": 11,

	"player_name": "Bob",
	"hole_cards": [
	{ "rank": "2", "suit": "hearts" },
	{ "rank": "7", "suit": "diamonds" }
	],
	"best_hand": "Pair",
	"hand_cards": [...],
	"is_winner": false
	}
	]
	}
	
	2. game:ended → all players in this game room
	{
	"game_id": 123,
	"winner_id": 10,
	"winner_name": "Alice",
	"winning_hand": "Straight Flush",
	"pot_awarded": 1200,
	"final_standings": [
	{
	"player_id": 10,
	"player_name": "Alice",
	"final_chips": 2150,
	"profit": 1150
	},
	{
	"player_id": 11,
	"player_name": "Bob",
	"final_chips": 0,
	"profit": -1000
	}
	],
	}
	
	3. game:chips:update → all players in this game room
	- Game phase now 'ended'
	- Pot reset to 0
	- All players' updated chip counts
	- Winner information
	- Final game statistics
	
	4. game:state:update → each player individually
	{
	"player_id": 10,
	"chips": 2150,
	"balance": 12150,
	"profit_this_hand": 1150
	}

	5. game:hand:complete → all players in this game room
	{
	“game_id”: 123,
	“message”: “Hand complete”,
	“ready_for_next_hand”: true,
	“players_remaining”: 2
	}

D. Leave Game **yuvraj**

HTTP Method & Route	Ex. POST /api/games/:game_id/leave
Purpose	Allows a player to leave an active or waiting poker game. If leaving during an active hand, the player is automatically folded. If chips remain, they return to the player’s user balance.If the player is the last one, the game will end.
Authorization	- User must be authenticated - User must be a player within the game (game participants) - Can be triggered by ANY player - Game must exist
Request Body	- none
Validation Checks	- The game exists - User is authenticated with the session - User is a participant of the game - If player is mid hand then fold
State Updates (What changes in the database?)	- Remove the player from game_participants. - Refund the player’s remaining chips back to their user balance. - If leaving during a hand → mark player as folded, remove from turn order. - If only one active player remains → award pot and end the game. - Update game’s remaining player count.

	- If the game ends → set game status to ended, record winner.
Success Response	- 202 Accepted
Error Cases (What can go wrong)	400 Bad Request - Game state invalid for leaving 401 Unauthorized - User is not logged in 403 Forbidden - User is not a player within the game 404 Not Found - Game does not exist 409 Conflict - Player already left

Socket.io Events (Which Events are triggered?)	5. game:player:left → sent to all players in this game room
	{
	"game_id": 123,
	"player_id": 10,
	"player_name": "Alice",
	"remaining_players": 3
	}

G. New hand **(Kataliya)**

HTTP Method & Route	Ex. POST /api/games/:game_id/new-hand
Purpose	This endpoint starts a brand new poker hand after the previous hand has ended. It moves the dealer button, shuffles and deals a new deck, post blinds, deals hole cards to all players, and begins a fresh betting round. This enables continuous multi-hand gameplay without creating an entirely new game.
Authorization	- User must be authenticated - User must be a player within the game (game participants) - Previous hand must be completely - At least 2 players must have chips remaining
Request Body	- none

Validation Checks	- The game exists - User is authenticated with the session - User is a participant of the game - User has chips remaining or is observing Specific Checks: - Game phase is ‘ended’ - At least 2 players have chips > 0 - No players are currently marked as ‘eliminated’ with chips > 0 - Game hasn’t been marked as permanently closed - All players are ready to continue

State Updates (What changes in the database?)	For all successful actions:
	- Log the showdown action in the actions table with game_id, action_type=’new_hand’, and system triggered flag
	
	1. Eliminate players with 0 chips
	   - Mark players with 0 chips as ‘eliminated’
	2. Move dealer button
	   - Rotate dealer button clockwise to next active player
	3. Reset game state for new hand
	   - Set phase to ‘pre-flop’
	   - Reset pot to 0
	   - Reset current bet to 0
	   - Increment hand_number
	4. Reset all active players’ betting state
	   - Clear all betting amounts and action flags
	   - Set all active players’ status to ‘active’
	   - Reset folded and all-in flags
	5. Create and shuffle deck
	   - Delete all cards from previous hand
	   - Create a fresh deck of 52 cards in game_cards table
	6. Post blinds
	   - Deduct small blind from player left of dealer, add to pot
	   - Deduct big blind from next player, add to pot

	- Set current_bet to big_blind amount 7. Deal Hole cards - Deal 2 cards to each active player from deck - Mark cards as 'player_hand' assignment to each player 8. Set First Player to Act - Set current_player_id to player left of big blind
Success Response	- 202 Accepted
Error Cases (What can go wrong)	400 Bad Request - Game is not in showdown phase and multiple players remain - Betting round is not complete - No active players remain 401 Unauthorized - User is not logged in 403 Forbidden - User is not a player within the game 404 Not Found - Game does not exist 409 Conflict - Game is not in a playing state

Socket.io Events (Which Events are triggered?)	1. game:hand:started → all players in this game room
	{
	"game_id": 123,
	"hand_number": 2,
	"dealer_button_position": 1,
	"pot_total": 30,
	"current_bet": 20,
	"current_player_id":10,
	"phase": "pre-flop"
	}
	
	2. game:hand:dealt → each player individually
	{
	"game_id": 123,
	"player_id": 10,
	"hole_cards": [
	{ "rank": "K", "suit": "hearts" },
	{ "rank": "Q", "suit": "diamonds" }
	]
	}

	3. game:state:update → all players in this game room - Updated public state, new hand number, dealer position, pot, current bet, whose turn, all players' chip counts 4. game:turn:notification → current player only - Notifies first player to act that its their turn with available actions