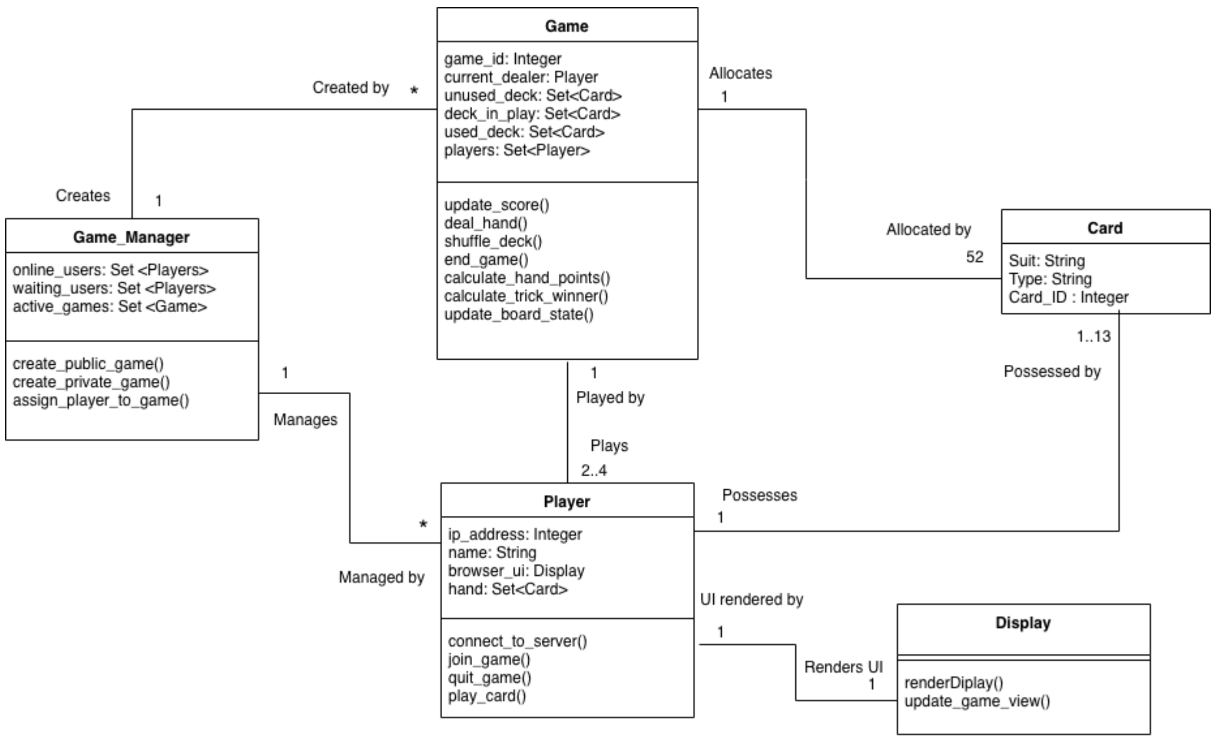
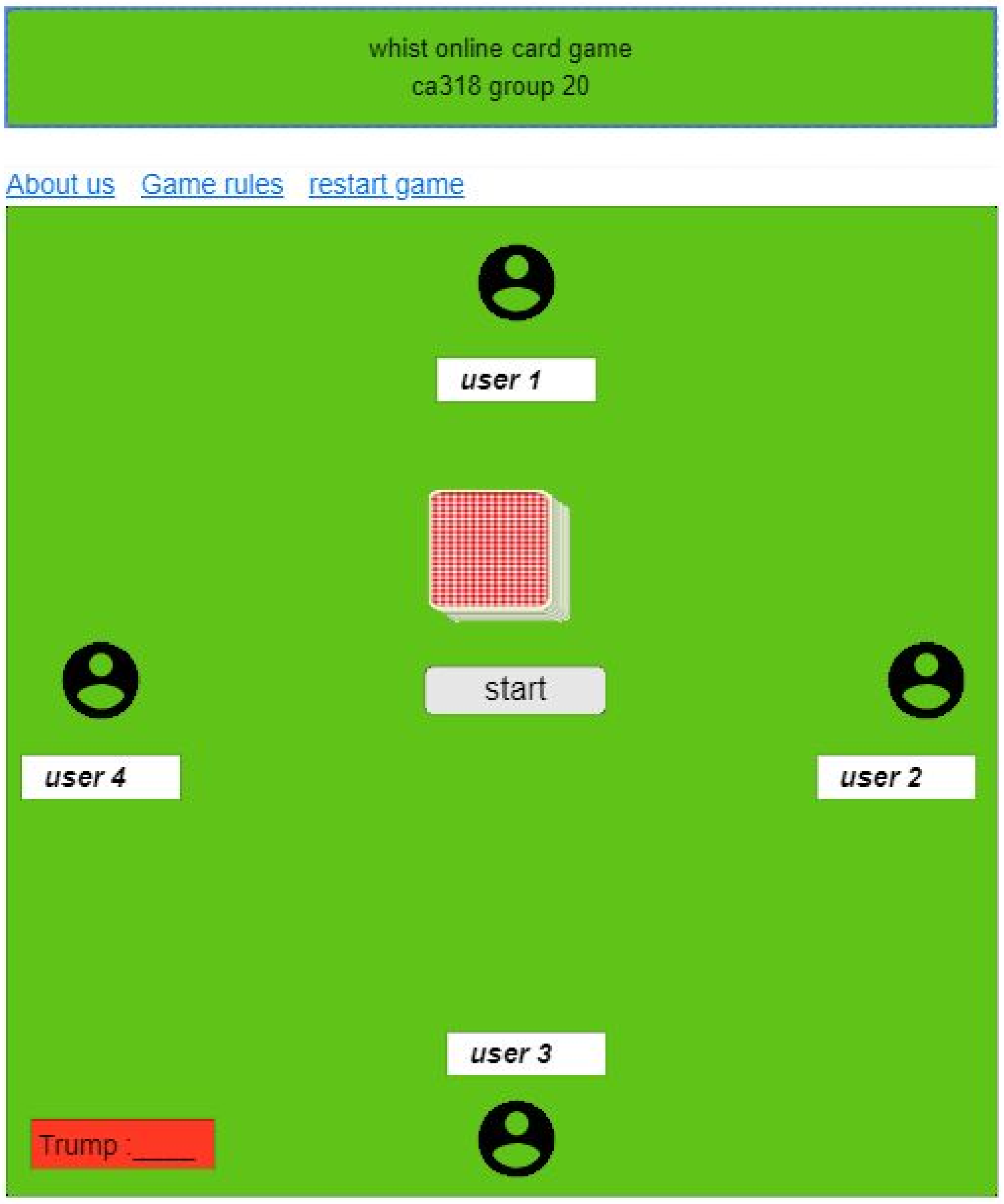
CA314 Assignment 2 - Product Design & Class Design

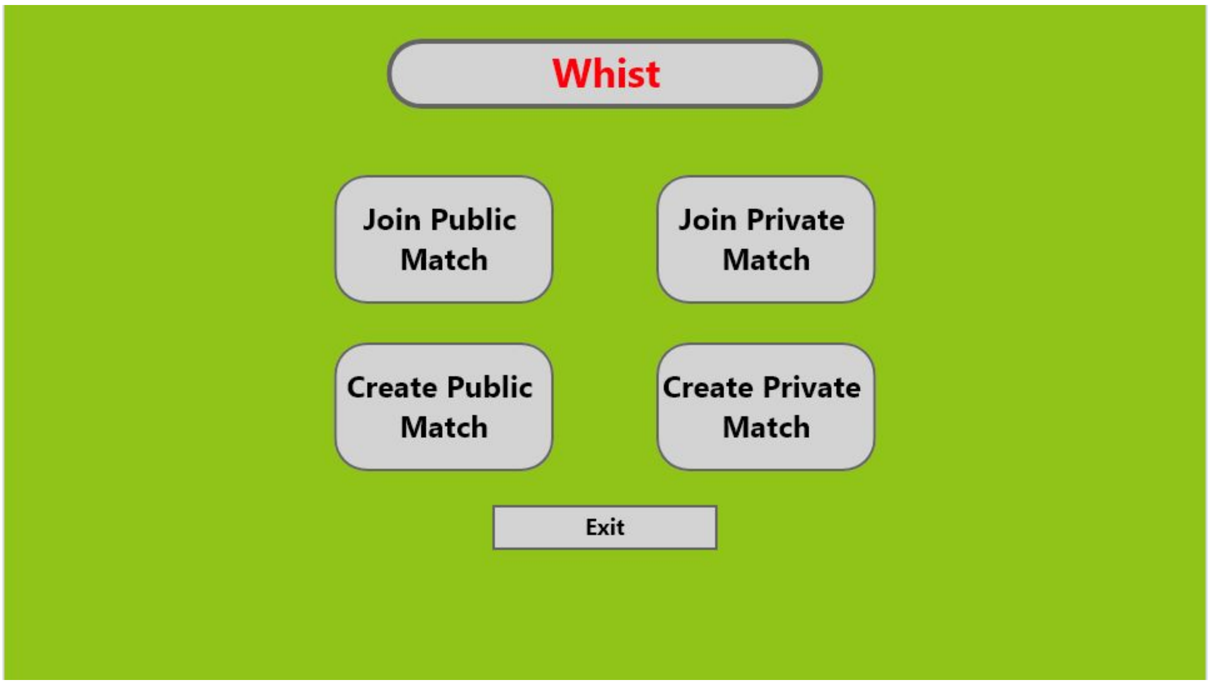
Group Name: Group 20
Members: Kieran Flynn, Walter Eze, James Toolen, Alaaldin Afana, Ting Lok Chang.
Game: Whist

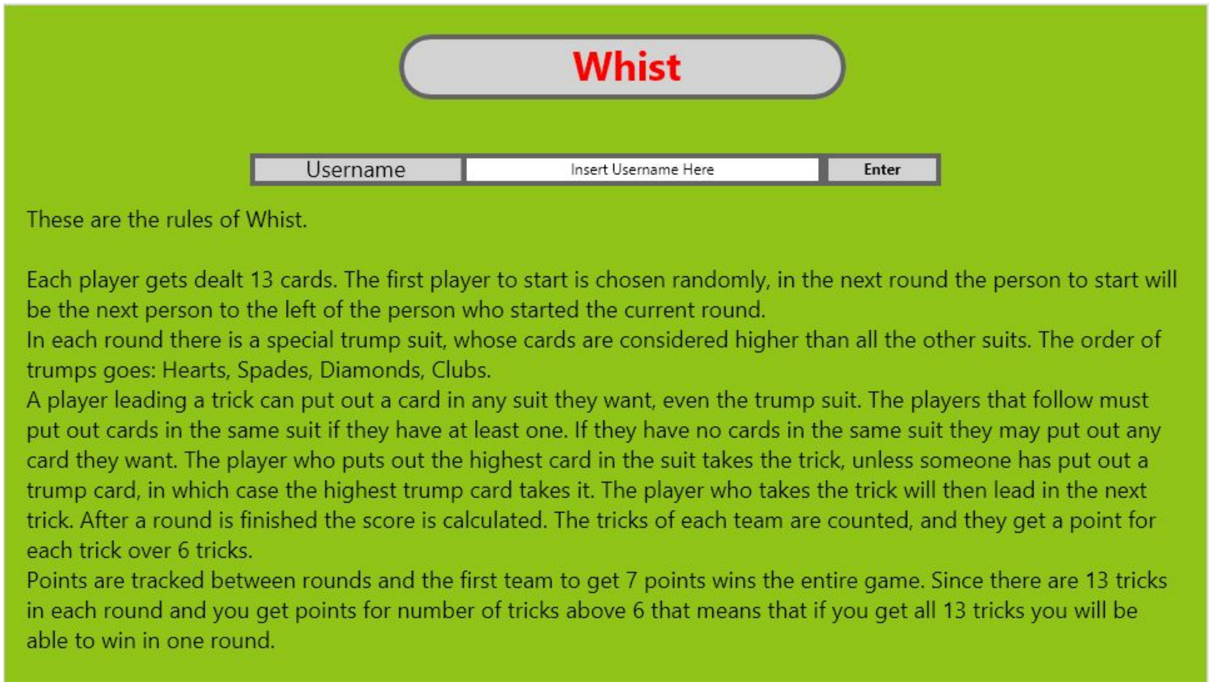
Redefined Class Diagram



User Interface Mock-Ups

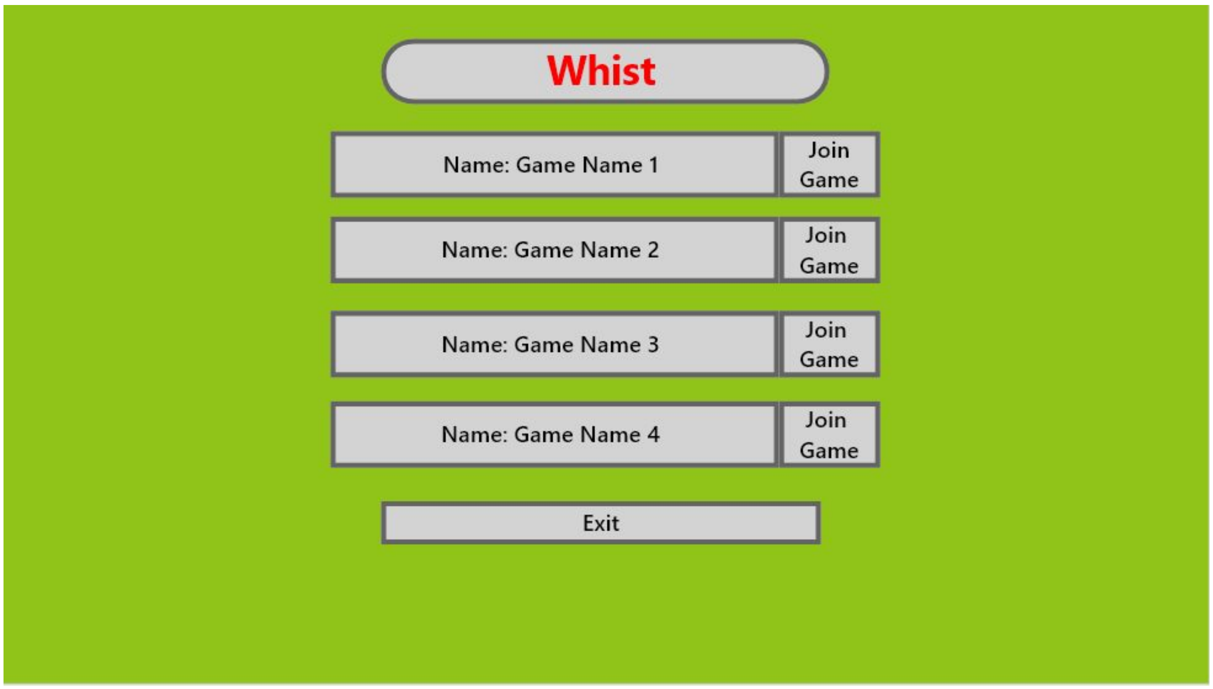


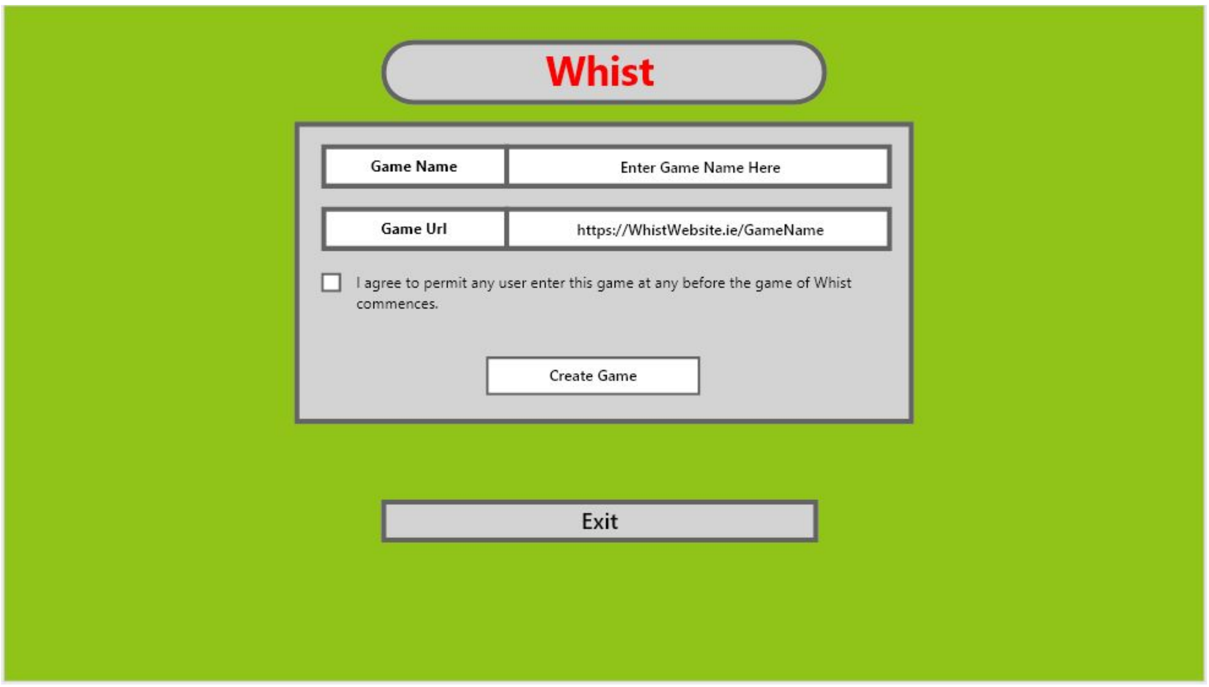




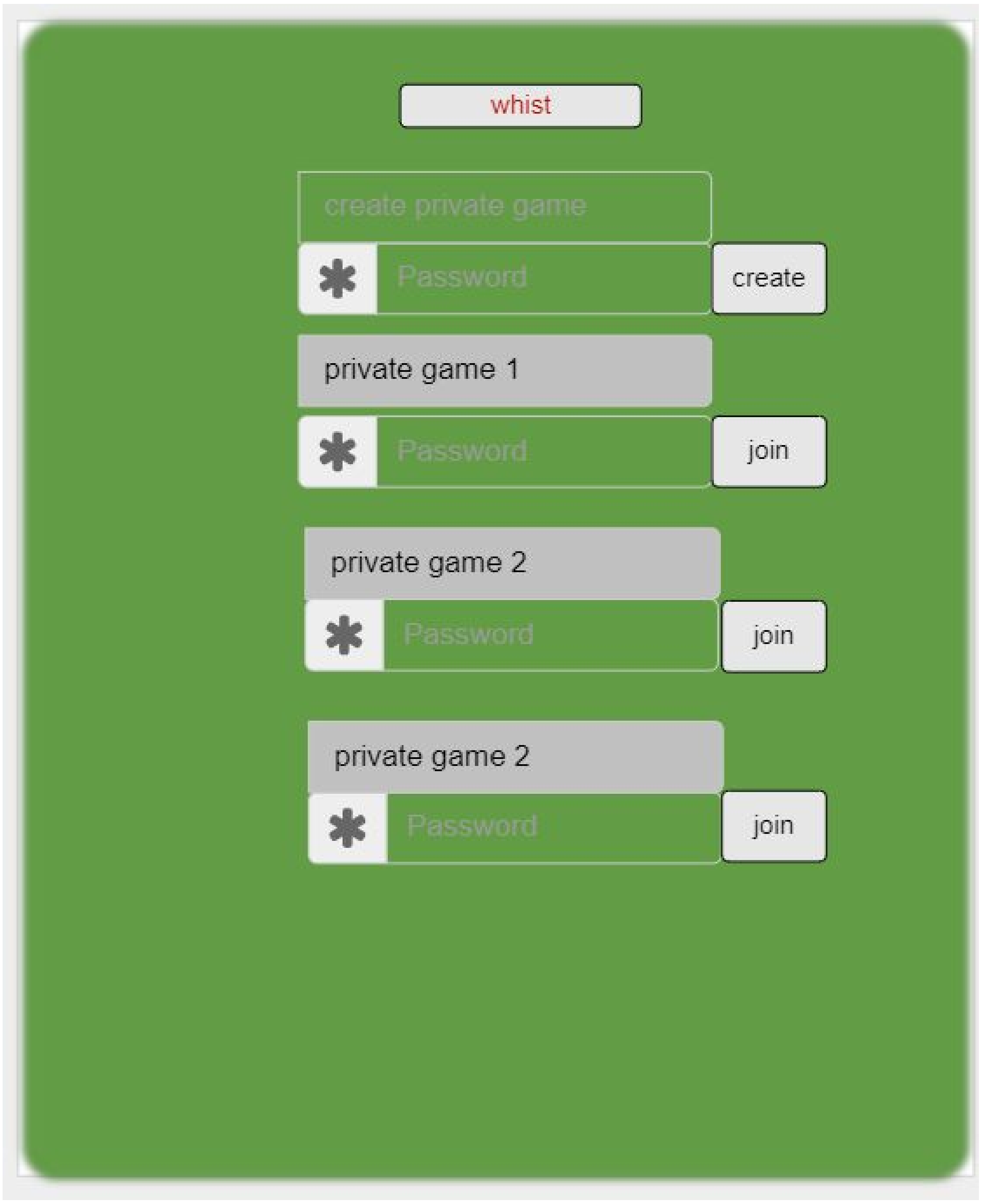
These are the rules of Whist.

Each player gets dealt 13 cards. The first player to start is chosen randomly, in the next round the person to start will be the next person to the left of the person who started the current round.
In each round there is a special trump suit, whose cards are considered higher than all the other suits. The order of trumps goes: Hearts, Spades, Diamonds, Clubs.
A player leading a trick can put out a card in any suit they want, even the trump suit. The players that follow must put out cards in the same suit if they have at least one. If they have no cards in the same suit they may put out any card they want. The player who puts out the highest card in the suit takes the trick, unless someone has put out a trump card, in which case the highest trump card takes it. The player who takes the trick will then lead in the next trick. After a round is finished the score is calculated. The tricks of each team are counted, and they get a point for each trick over 6 tricks.
Points are tracked between rounds and the first team to get 7 points wins the entire game. Since there are 13 tricks in each round and you get points for number of tricks above 6 that means that if you get all 13 tricks you will be able to win in one round.

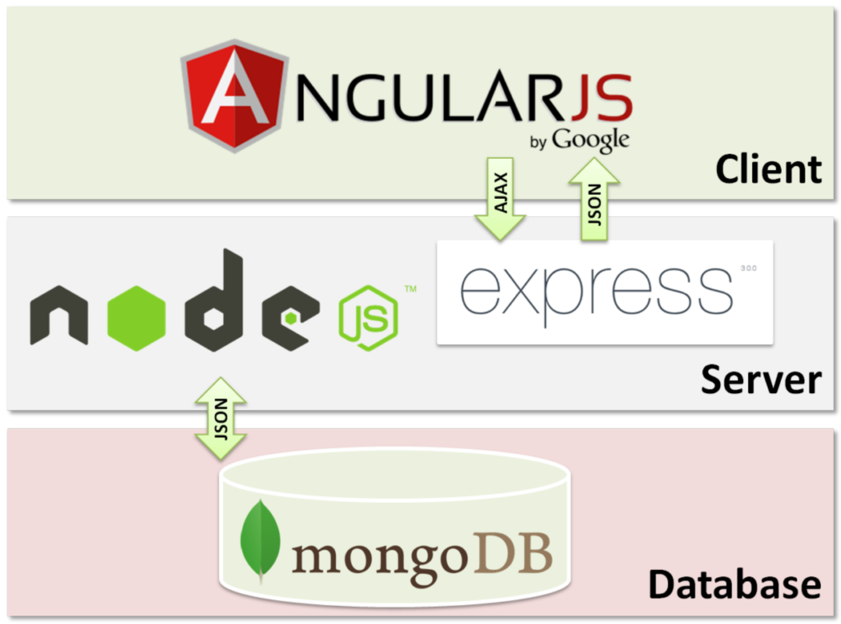




☐ I agree to permit any user enter this game at any before the game of Whist commences.


whist
create private game
Password
create
private game 1
Password
join
private game 2
Password
join
private game 2
Password
join

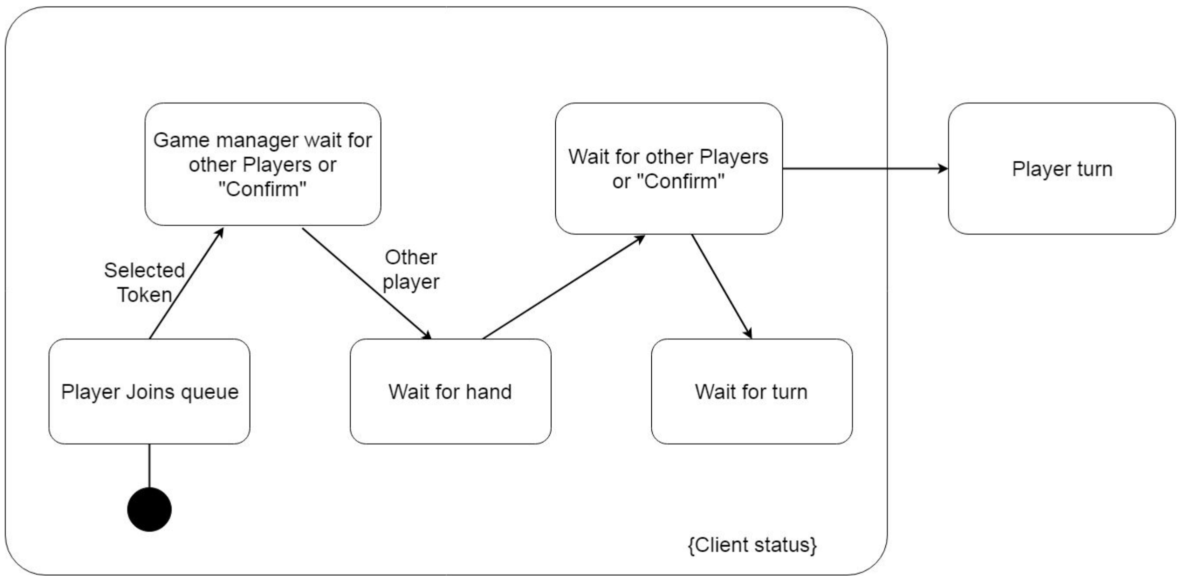
Client Server Experiments / Chosen Architecture

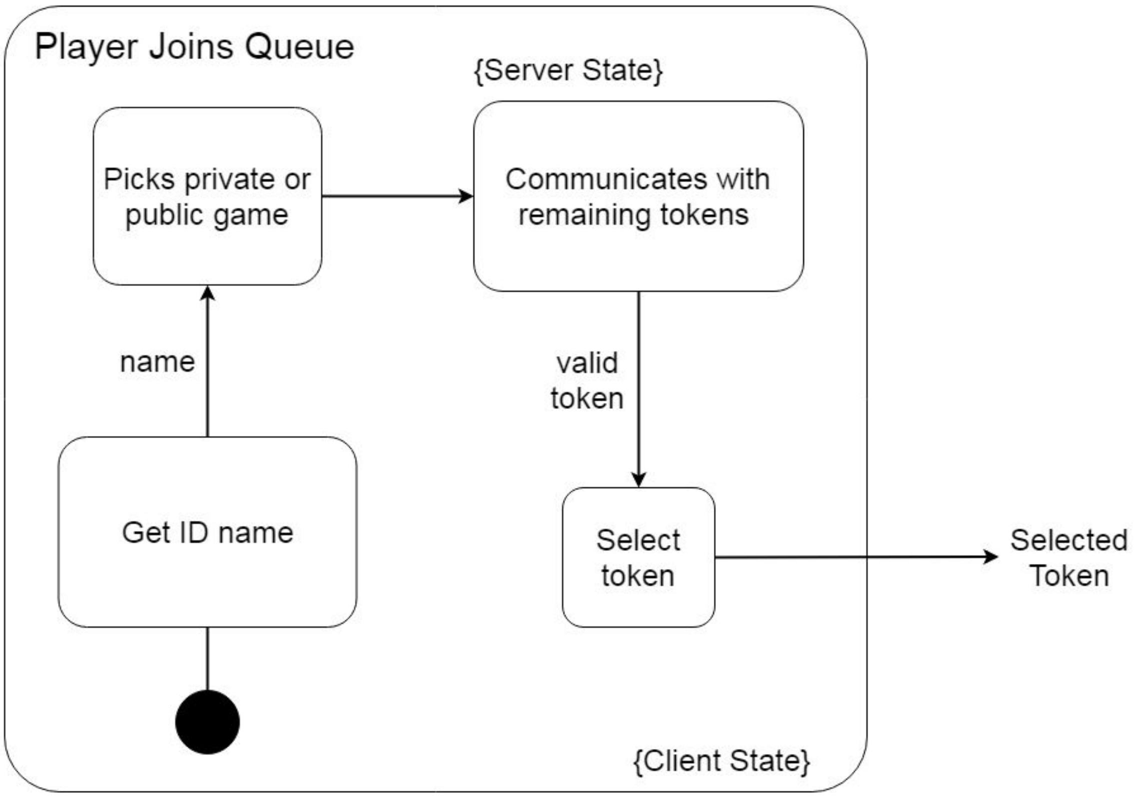


The client-server architecture we have chosen for implementation of this project is the MEAN stack. MEAN is a free and open source JavaScript based web development stack. As all elements of the stack support JavaScript it means the entire application can be written using a single language and will minimise integration problems and enable a very quick set up time as opposed to other architectures. The nature and complexity of our proposed system indicates that the MEAN stack will easily be able to support our Whist web application.

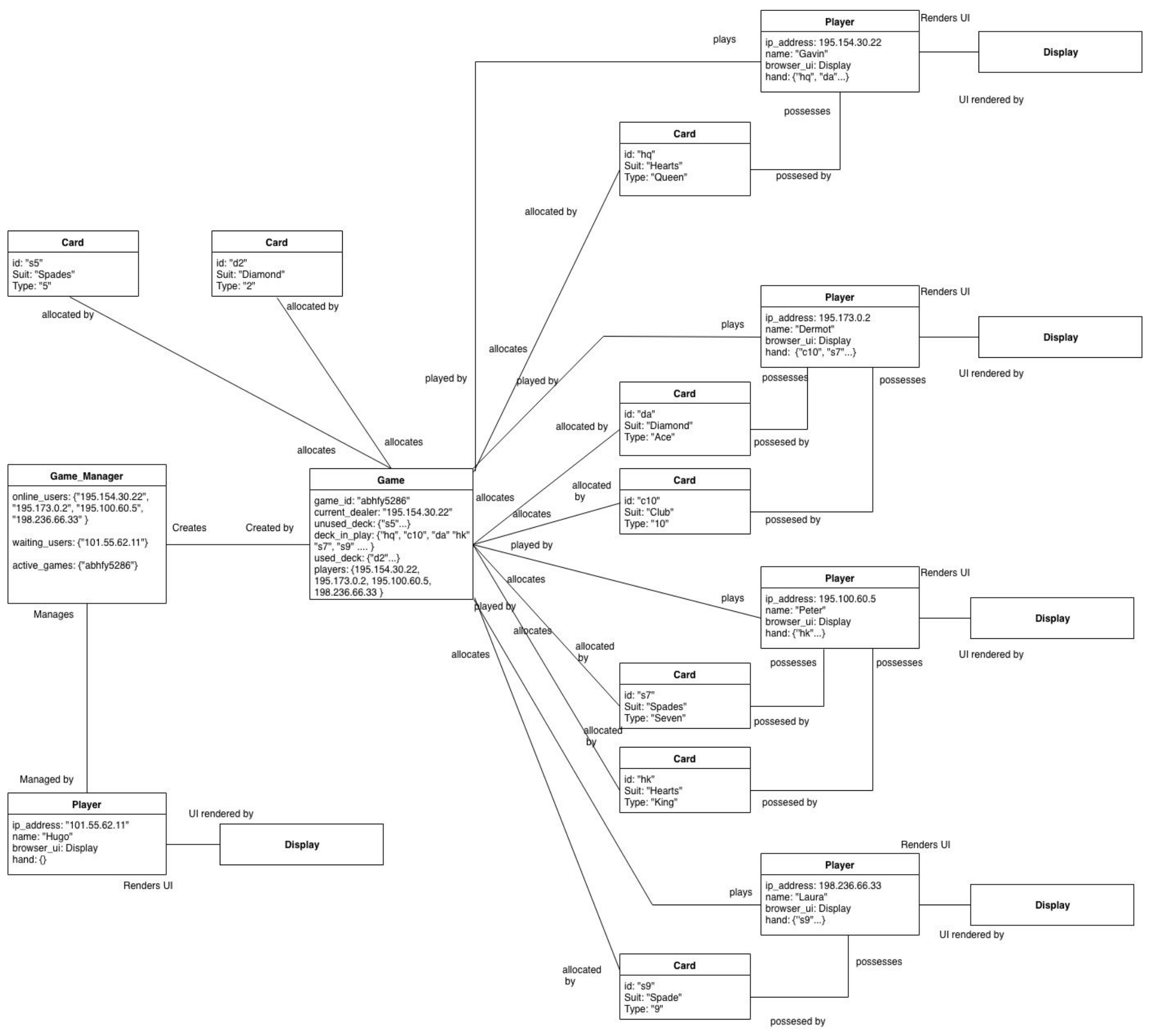
State Diagrams

State Machine for initiating a Game Client[Client-Server]



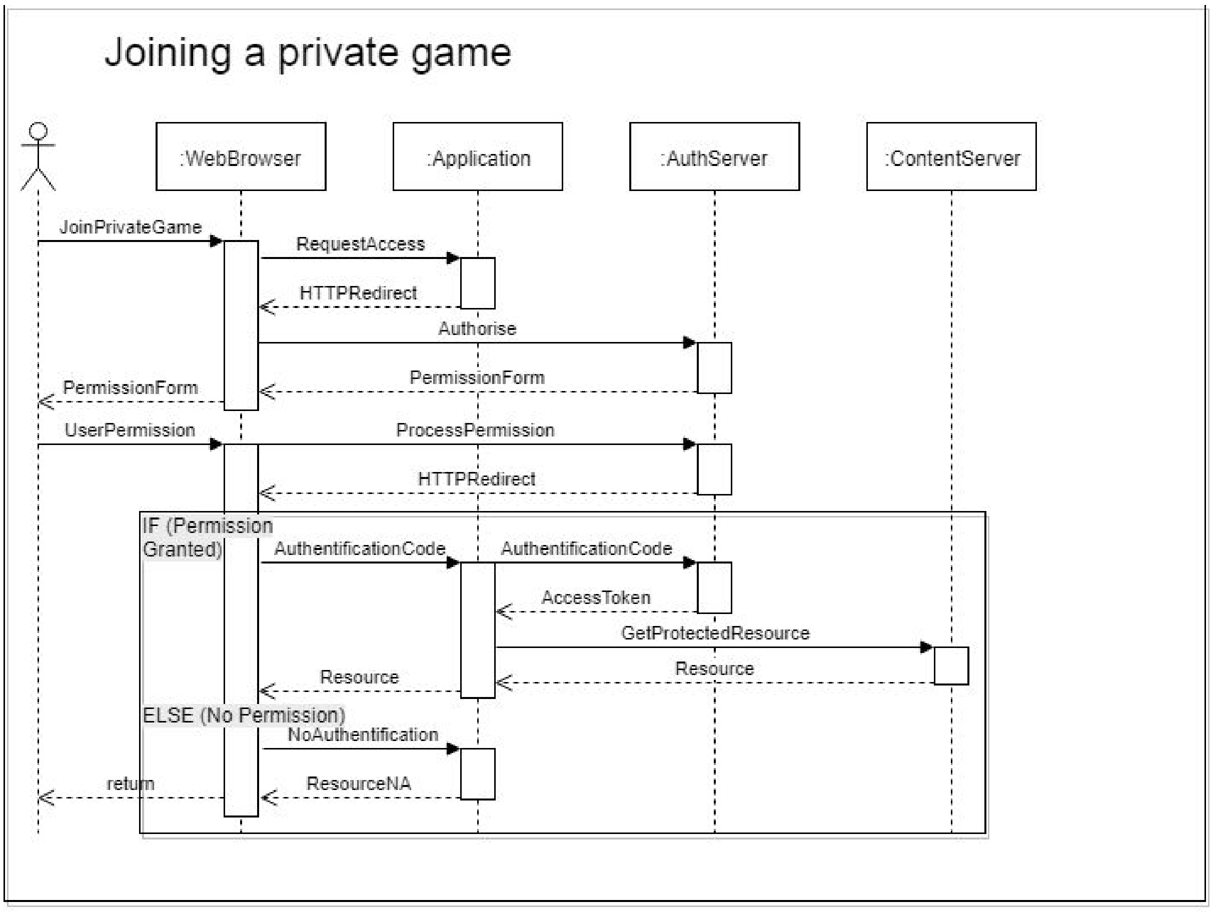


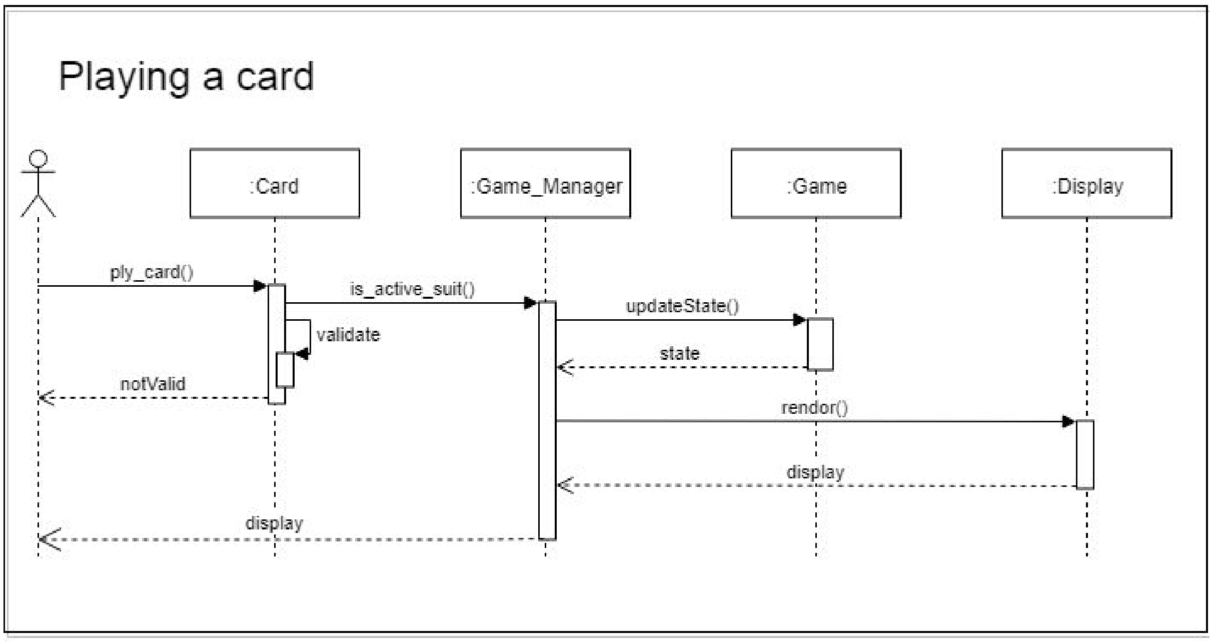
Object Diagrams



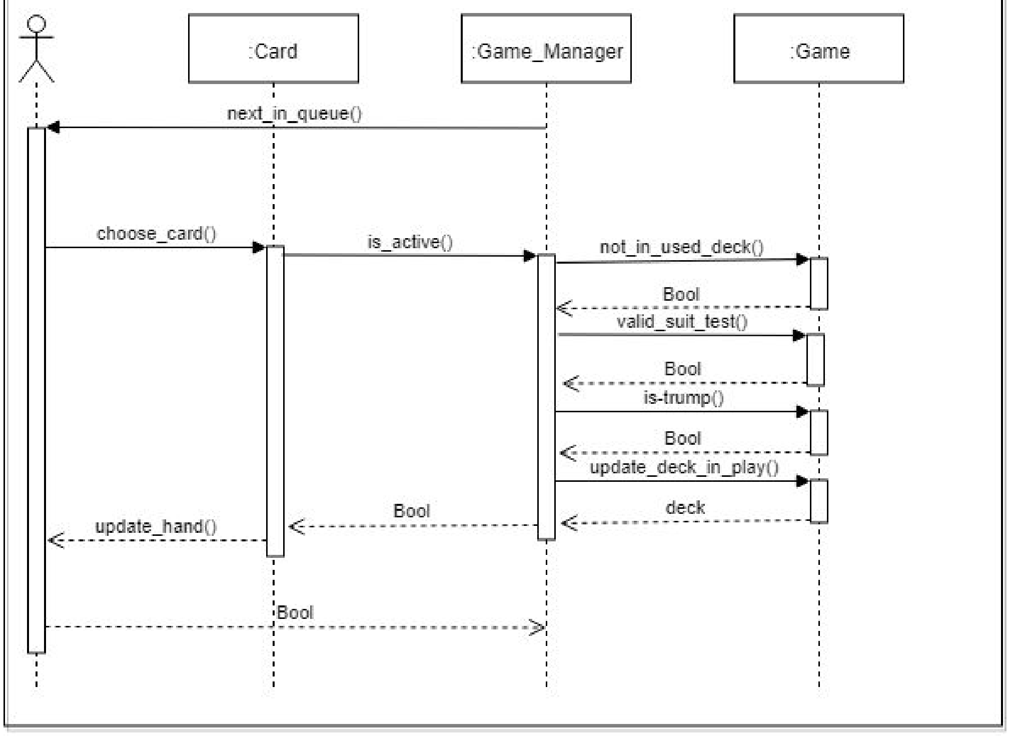
Note: We have only shown 8 card objects in this diagram in order to keep it as clean as possible. This snapshot displays one active game, each active game has 52 card objects associated with it, only 8 of the objects associated with this game are shown

Sequence Diagrams

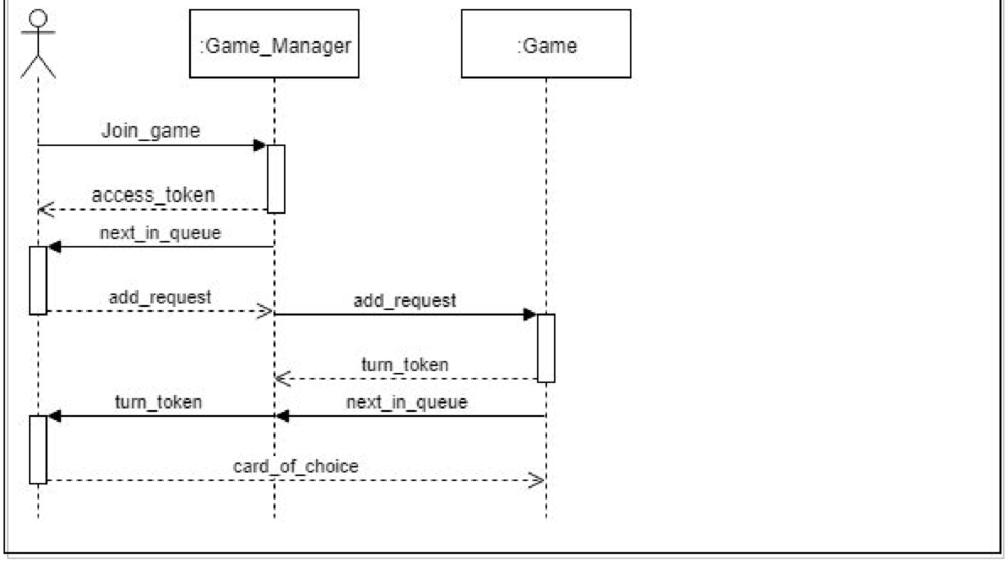




Calling next player

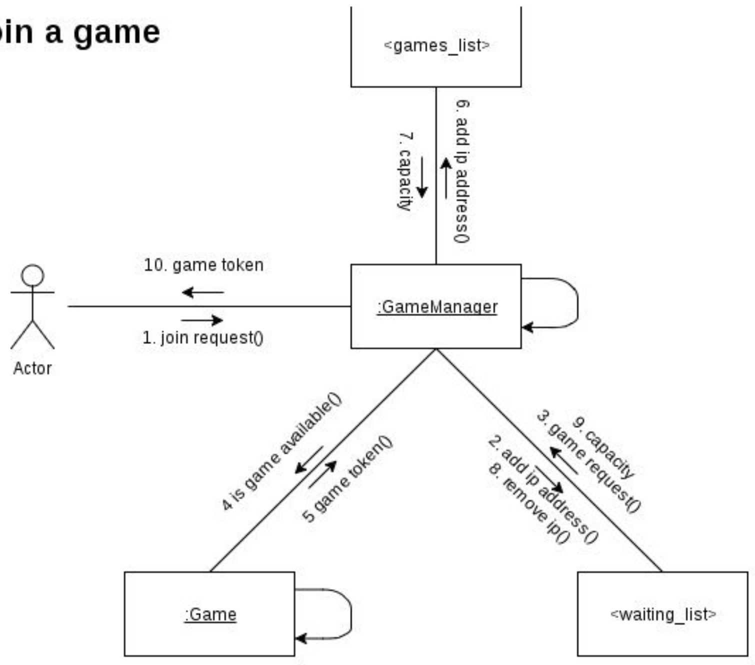


Joining a random game

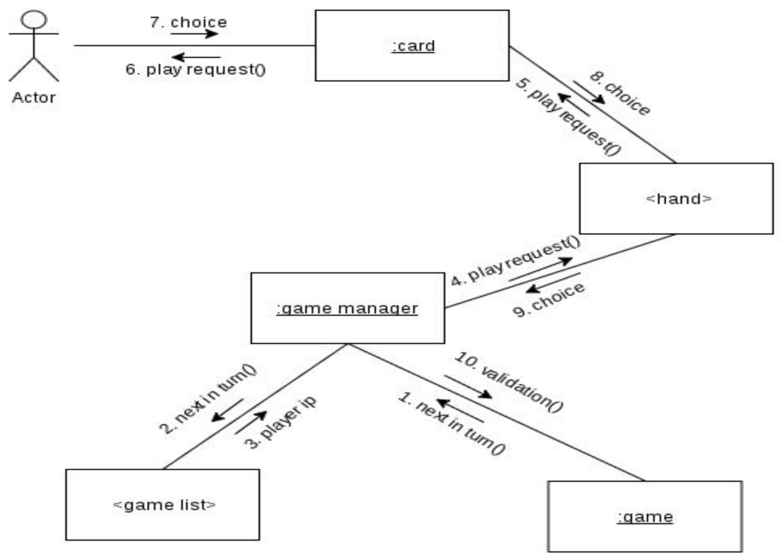


Collaboration diagrams

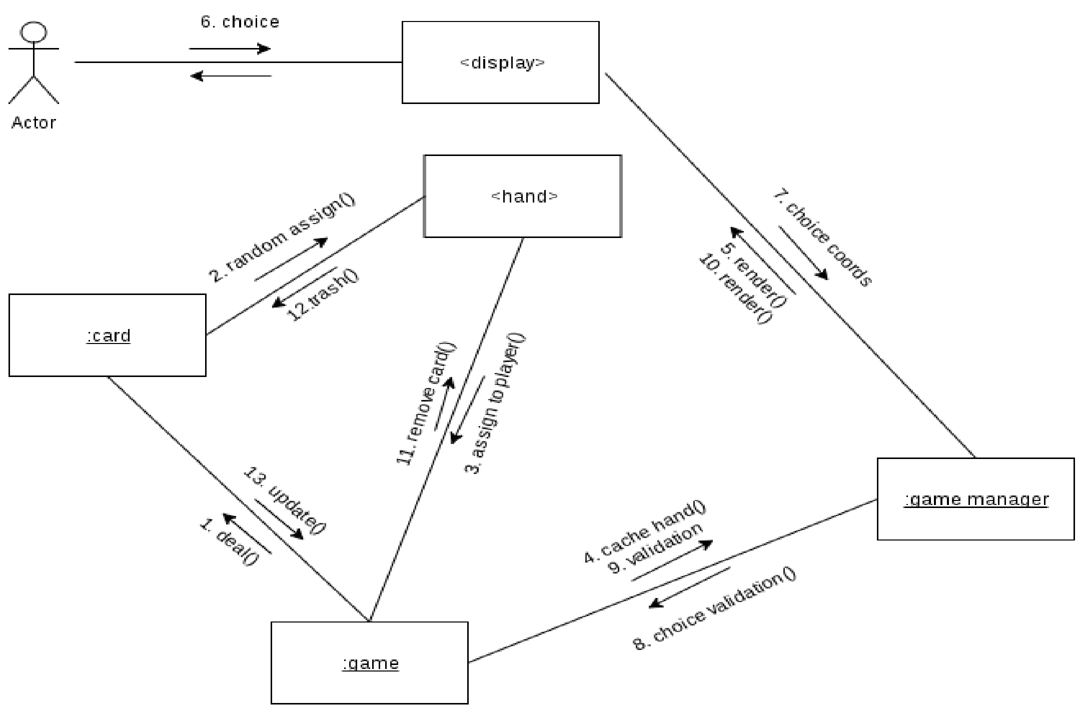
Join a game



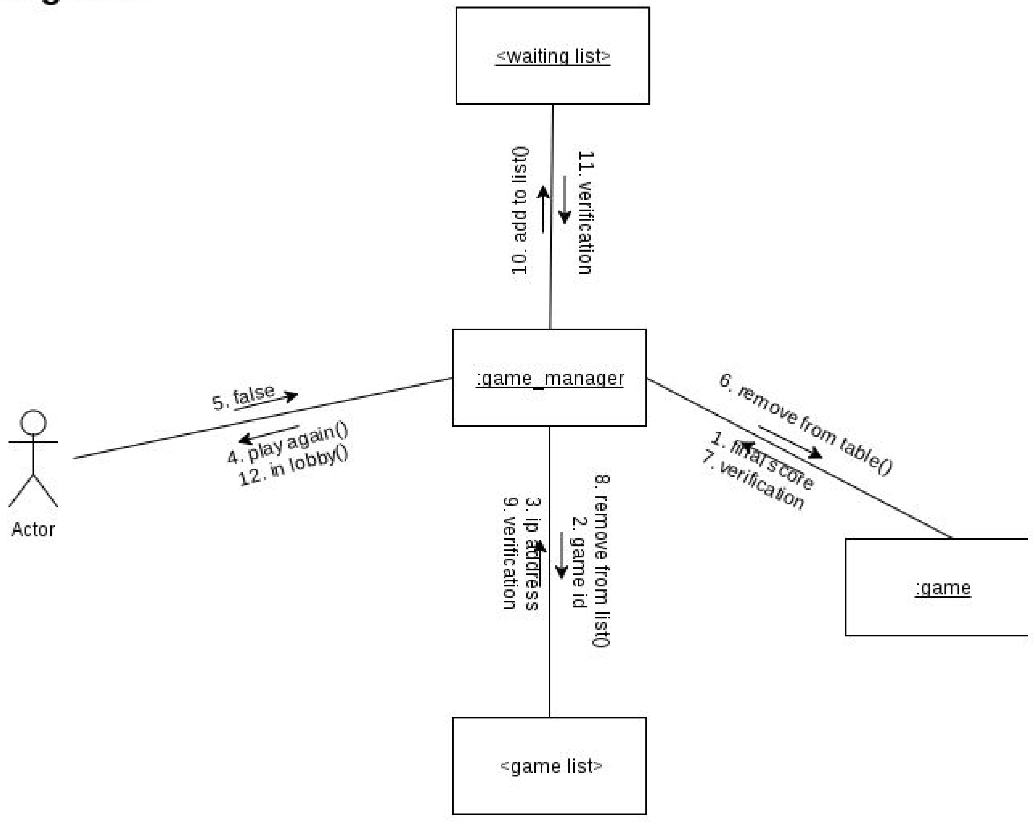
Play a card



Display



End game



Class Skeleton

```
/**
 * Description of Game_manager.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
class Game_Manager{
        online_users: Set <Players>
        //keeping in check of how many players are available
        waiting_users: Set <Players>
        //tracking players that wants to join a game
        active_games: Set <Games>
        //tracking numbers of game that had players in it

        create_public_game()
        //method to initialise a public game
        create_private_games()
        //method to initialise a private game
        assign_player_to_game()
        //method to place waiting players into their choice of game
}

class Game{
        game_id: Integer
        //ID of the game, e.g: game 123456
        trashed_deck: Set <Card>
        //track cards not in play
        deck_in_play: Set <Card>
        //tracking cards that is dealt to players and on the field
        players: Set <Players>
        //2 to 4 players

        update_score()
        //update players' scores
        deal_hand()
        //cards dealt to each player

        shuffle_deck()
        //randomising 52 individual cards in deck
        calculate_hand_points()
        //keeping players' hand points
        calculate_trick_winner()
        //keeping players' score
        update_board_state()
        //function to update what everyone can sees when a card is played
        end_game()
        //terminate other in-game functions after winner is declared
}

class Player{
        ip_address: Integer
        name: String
        browser_ui: Display
        hand: Set <Card>
        //class Players holds ip address info of players,
        //their unqiue id, name and UI shows thier hand
        //hand is given a set of 13 cards to be used

        connect_to_server()
        //connects to server
        join_game()
        //obtain token from server to join game
        quit_game()
        //release token to leave game
        play_card()
        //play card in hand

}

class Card{
        Suit: String
        Type: String
        Card_ID: Integer

        //class card holds the data of a single card(allocated by 52 of this) of its suit & type &
card ID number
}



class Display{
        renderDisplay()
        update_game_view()
        //card display shows what info player can see on their screen
}
```

Team Meeting Minutes

CA314 Project – Meeting Three's Minutes

Date: 31st October 2018

Attendees: James Toolen, Kieran Flynn, Walter Eze, Ting Lok Chang

Meeting Results:

1. We debated our approach to the first stage of the project & what we can learn to do better in the next stages
2. We agreed that there was issues with communication of changes to the diagrams that affected the information for other people's work & we aim to improve our communication in future.
3. We agree that work could be more evenly distributed amongst all members & we aim to have more time at meetings dedicated to assigning work.
4. We discussed what sections each of us would like to pursue for the product design section. Ting will work on the State Machines, Kieran will work on the user interface mock-ups & will ask Alaaldin to assist, Walter will work on Object Diagrams, James will work on the Refined Class Diagrams & will ask Alaaldin to assist.
5. We set the agenda for the next meeting for allocating roles for the class design sections.

CA314 Meeting Four's Minutes

Date: 7th November 2018

Attendees: James Toolen, Walter Eze, Kieran Flynn, Alaaldin Afana, Ting Lok Chang

Results of Meeting:

1. We showcase our work for the product design section & agree that each section is of high enough standard to proceed to the class design section.
2. We discuss which sections each of us would like to work on for the class design section.
3. James & Kieran will work on the Collaboration diagrams.
4. James will work on the Sequence diagrams.
5. Walter will work on the Class diagrams & the revised Object diagrams.
6. Alaaldin & Ting will work on the Class skeletons.
7. Alaaldin will work on the Client Server implementation but everyone will research into resources to help create them as everyone seems to know little about them.
8. Kieran will continue to handle the team minutes.