Participants
==============

Dor Azouri   204034300 dorazouri@gmail.com
Moran Mahabi 307994491 moran.mahabi@gmail.com

General
==============

Our design model separates model methods from UI methods.
It is using prototypes as the way to implement JS classes.
The JS classes form the "model" part of the application, where all logic is implemented and all game entities are represented.
CSS classes are defined to allow better definition and interaction with the UI.
The card images are named according to a defined convention that allows the "binding" of a card image file to its model entity.

Classes
==============

1. Game - this is the main class, responsible for the orchestration of all the other models, and the flow of the game entities.
   This acts as the root object and is bound to the DOM's 'window' property to allow its persistency and easy access to it in
   methods that interact with the UI.
2. Card - Mostly self explanatory. Has a method that generates the corresponding image file name, to allow binding of a Card
   class to its image file.
3. Player - Represents a player of the game, both a human player and a computer player. The decision to differentiate the two
   using a class property (used as a flag), rather then using inheritance and polymorphism - has been made because this kind
   of implementation complexity is not required.
4. Deck - represents the deck of cards from which the players can take cards. Acts as a stack.
5. OpenDeck - represents the main heap of cards, on which the players put cards.
6. MoveGenerator - a helper class that holds the logic for the computer player.
7. UndoCaretaker - this is the caretaker part of the memento design pattern of undo and redo
8. UndoFrame - this is the memento part of the memento design pattern for undo and redo, holds the state before the undo or redo

Components
==============

1. In general, classes in the .*Comp format are react components.
2. GameComp - The main component, renders the different sub-components and passes the game model object to propagate down to the lowest component in the tree.
3. StatusBarComp - Corresponds to the status bar line drawable at the top of the page
4. BoardComp - Corresponds to the main part of the page, composed of the DecksComp at the top, and the PlayerComps beneath it
5. ChangeColorComp - Corresponds to the choose color palette, drawn when user chooses the changeColor card

6. EndGameStatisticsComp - Corresponds to the statistics modal shown at end of game
7. PlayerStatisticsRowComp - Corresponds to one line of the statistics table drawn in the EndGameStatisticsComp
8. DecksComp - Corresponds to the part at the top of the BoardComp, that draws the open deck and closed deck of cards. Conditional drawing, according to whether it is an open deck or a regular one
9. PlayerComp - Corresponds to the horizontal componenets that holds all the CardComps a player has in its hand. Holds some logic of how to draw the cards, based on the player type and the turn in game
10. CardComp - Corresponds to the basic card drawable, shown at the players' hands. Top level logic of the onClick handling is defined in it

Assumptions
==============
1. The game is for 1 computer player, and 1 human player.

Remarks
==============
* We chose to help the user understand the possible cards he can click, by changing the mouse pointer to look clickable.