Software Components:

1) Create Dataframe

What it does: Create Dataframe component takes in ~30,000 raw text files that contain various amounts of poker games played on 888poker.com during heads up Texas Hold'em. It contains many functions that splits the raw data into keys and values such as 'Flop actions', 'Flop', 'BB cards', 'Folded pre', and many more. These keys and values are then stored as a pandas dataframe, with each row representing a full game played.

Input: ~30,000 raw text files (.txt) stored in Peter's computer

Output: pandas dataframe of 520,810 rows (games) and 16 columns (different game metrics and values)

2) Evaluator

What it does: Evaluator component is used to evaluate a player's two card hand and the board, either in 3 (flop), 4 (turn), or 5 (river) card situations and return the rank class (e.g. pair, straight, full house, etc.) and rank percentage (between 0 and 1), with 1 being the best possible poker hand, a royal flush, and 0 being the worst possible poker hand, unsuited 7-5-4-3-2. Using card.py, card inputs such as Ts (ten of spades), are converted to a unique set of 32 bits used in manipulation and comparison. These cards, in bitwise form, are then used to set the player's hand and the board, which are used by Evaluator to output the player's rank class and rank percentage. The evaluator analyzes each row (game) in the dataframe and outputs new columns that are added to the dataframeThese are used by other components for further analysis and as a metric to train our ML algorithm

Input: hand and board, in the form of:

```
hand1 = [
   Card.new('Ts'),
   Card.new('Ac')

board = [
   Card.new('2h'),
   Card.new('2s'),
   Card.new('Jc'),
   Card.new('Kh'),
   Card.new('Qc')
```

The inputs are the cards of each player and board from each game in the dataframe with each variable stored as lists.

Output: rank classes and rank percentages for each player per game are added as columns (‘SB Handrank’, ‘BB Handrank’, ‘SB Hand Strength’, ‘BB Hand Strength’) into the pandas dataframe.

Ex of calculations per game)
rankclass1 = evaluator.class_to_string(evaluator.get_rank_class(evaluator.evaluate(hand1, board)))
rankperc1 = evaluator.get_rank_percentage(evaluator.evaluate(hand1, board))
rankclass1 = Straight
rankperc1 = 0.785580

3) Adding Flush

What it does: Adding Flush component only analyzes games that have made it to the river and classifies the likelihood of a flush on the board. Filtering so that only these games are included, a new dataframe called pk_rvr is created. It creates a new column in this dataframe called ‘Board’ which is the combination of the ‘Flop’, ‘Turn’, and ‘River’ columns and calculates the highest number of the same suit of the board. Based on this highest value, ‘Flush Count’ is added to the dataframe. Then based on this flush count value, a new column is created called ‘Flush’ which takes the value of 0, 1, 2, or 3. The conversion between ‘Flush Count’ and ‘Flush’ is as follows;

Flush Count	Flush
1 or 2	0
3	1
4	2
5	3

Input: A csv file of the original data frame made in Create Dataframe component

Output: A new data frame called pk_rvr**,** which is a subset of the original data frame, with added columns ‘Flush Count’ and ‘Flush’

Interactions to accomplish use cases:

Use Case 1: User Interaction

The user will input their cards, board, and previous personal and opponent actions during the river stage of the game. These cards will be run through Evaluator to get the rank class and rank percentage of their hand. Once this is done, our ML algorithm, which has been trained using the data frames created in Create Dataframe and Adding Flush as well as many more, will analyze the players hand, previous rounds actions, and compare the current game to the hundred of thousands of games stored in our data frames to give insights into the likelihood of winning against a random opponent, as well as recommended actions.

Preliminary Plan: