Node Dealer

Using the MEAN stack, create a single-page web app that shuffles a standard deck of 52 cards, deals them in a matrix of 13 columns by 4 rows, and calculates the deal's "score" based on the rules to be described.

For each deal, store the deck configuration and the score in the database. Display an average score across all historical scores. Display the deck with a button labeled "Deal Again" to deal the next deck.

Deliver us the complete source code for your working application. We'd appreciate being able to run/use it with a minimum of setup and fuss, so please consider this as you decide how you'd like to deliver it.

Scoring rules:

Each of the four rows in the matrix is home to one suit — from top to bottom: spades, diamonds, hearts, then clubs. Each card whose suit matches its row is worth one point, i.e. each spade that appears in the spade row is worth one point; each diamond that appears in the diamond row is worth one point, etc.

Likewise, each of the thirteen columns in the matrix is home to one face value, ace through king. Each card whose face value matches its column is worth one point as well.

Thus the (very unlikely) highest-scoring configuration possible would look like this:

```
    A  2  3  4  5  6  7  8  9  T  J  Q  K
(S)[A][2][3][4][5][6][7][8][9][T][J][Q][K]
(D)[A][2][3][4][5][6][7][8][9][T][J][Q][K]
(H)[A][2][3][4][5][6][7][8][9][T][J][Q][K]
(C)[A][2][3][4][5][6][7][8][9][T][J][Q][K]
```

Express a configuration's score as the ratio of points it received out of the total possible points for an ideal deck (or in other words, as a percentage). The above configuration would have a score of 100%, because it received 104 out of 104 points.

Technical guidelines:

Node.js: use Node to serve the application.
MongoDB: use MongoDB as the persistent datastore to save the configurations and percentages. Use mongoose.js for the data access layer.
Express.js: use the Express framework to organize the server-side code and handle url routing.
Angular.js: use Angular as the front-end templating engine.

Bonuses:

1.) Add a UI to access and display historical decks, honoring the order in which the decks were generated.
2.) Highlight adjacent pairs, triplets, and quads (horizontal, vertical, or diagonal).