**README-**

**Note:**

The compilation and deployment sections are provided with instructions for Linux/MAC command line. For windows, please use respective windows command line options.

**Deployment Steps:**

Execute the following in command line.

1. Extract Archive

2. Change to “MotifUIapp” directory

cd MotifUIapp

**Prerequisites:**

**Install Node.js**

**To Build the project:**

1. Install Node packages:

npm install

2. Install Bower:

npm install bower

3. Install Bower packages:

bower install

**To Run the project:**

grunt serve --force

**Working of the Application.**

**1st**

Here I am taking the input from the user. The user can select the number of Features (rows) and the number of Privacy Level (columns) from the input.

After submitting the table will be populated. To populate the data, I have written a function $scope.makeAnList which upon submitting the Features (rows) and Privacy Level (columns) generates JSON data.

$scope.list = {

"Feature 1": [

{ id: 1, selected: false, level: 1 },

{ id: 2, selected: false, level: 1 },

{ id: 3, selected: false, level: 1 },

],

"Feature 2": [

{ id: 1, selected: false, level: 2 },

{ id: 2, selected: false, level: 2 },

{ id: 3, selected: false, level: 2 },

],

"Feature 3": [

{ id: 1, selected: false, level: 3 },

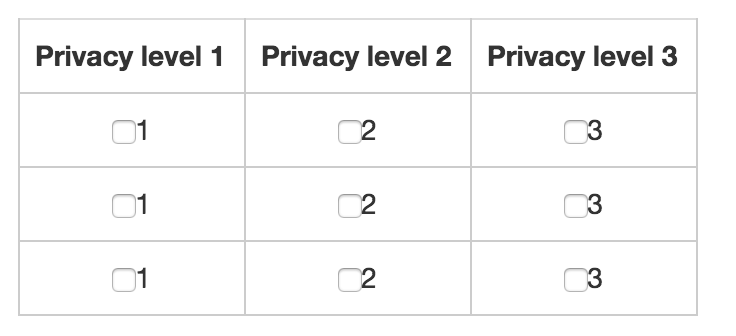
{ id: 2, selected: false, level: 3 },

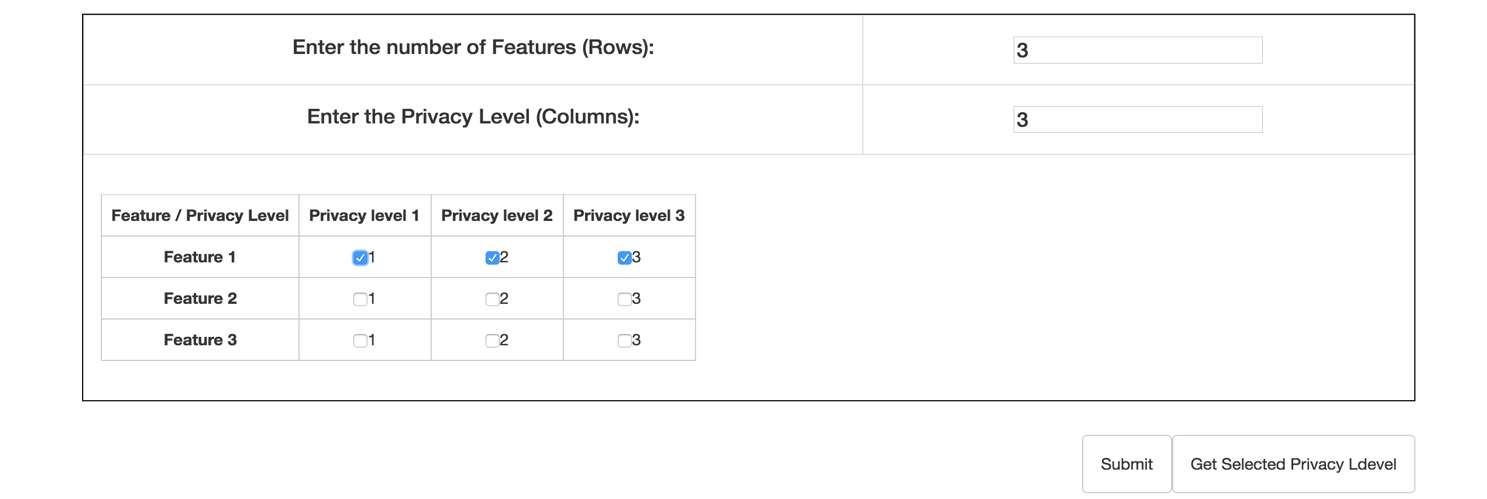
{ id: 3, selected: false, level: 3 },

]

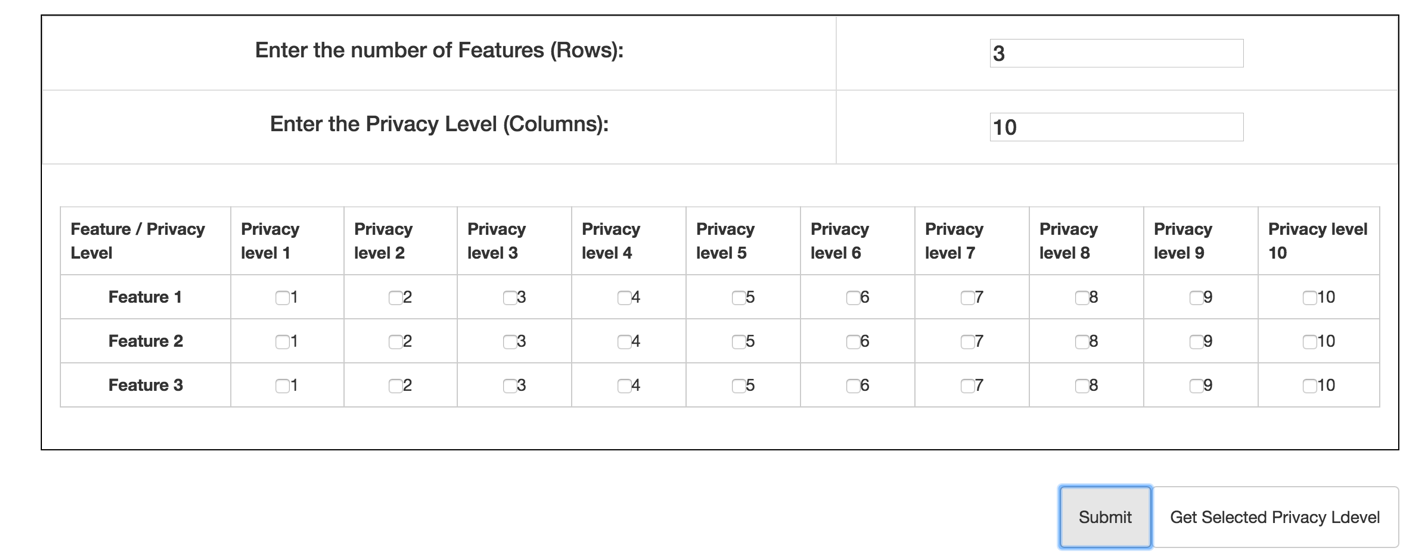
};

The above JSON file is representing.





**For 3 Features and 3 Privacy Level**



**For 3 Features and 10 Privacy Level**

The $scope.getSelectedCheckboxesToTheConsole prints the selected checkboxes and its respective privacy level to the console.

The $scope.isChecked function satisfies the required 2 conditions of checkbox.

The $scope.getNumber function generates 2 arrays for Privacy level and Feature for table name incrementing.

**2nd**

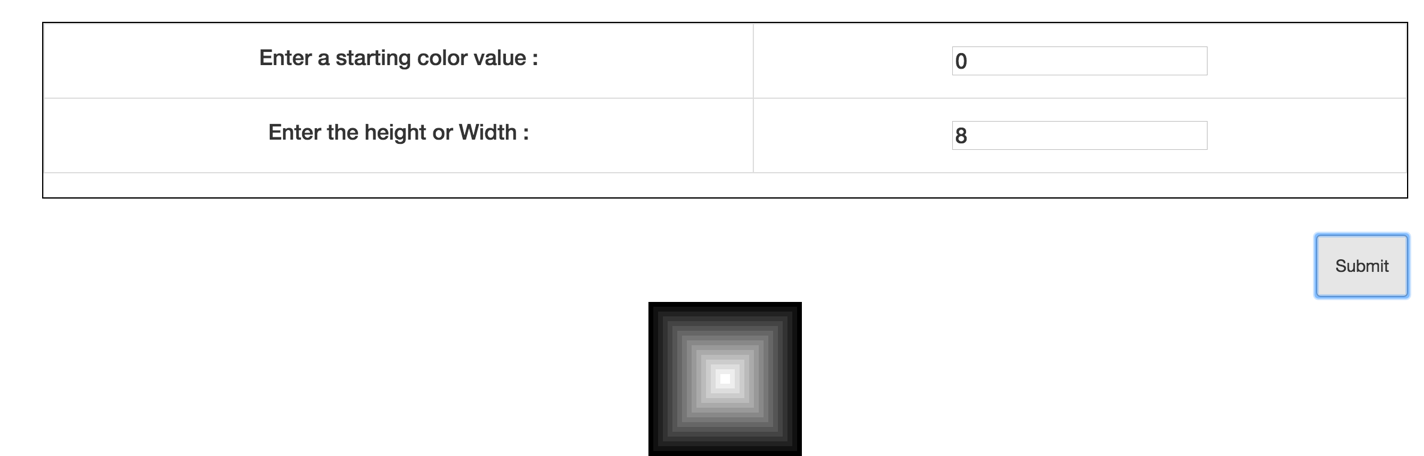
Here there is single function $scope.drawSquares which generates the required images.

Though I misunderstood the requirements at first place, the code is very straightforward.

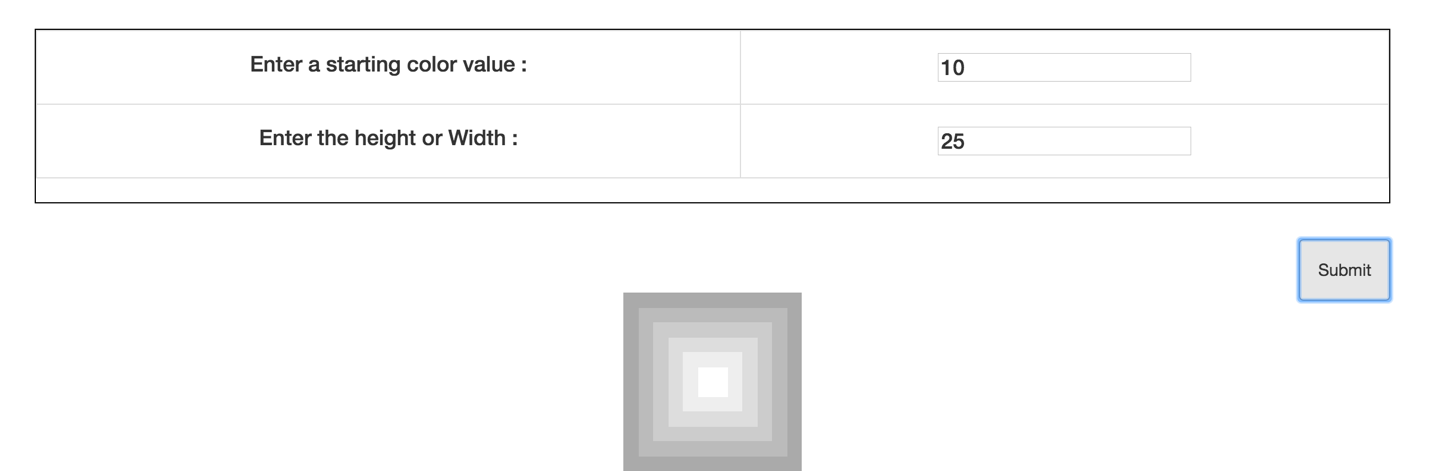
It takes the input as decimal and generates an object which is nothing but a gray scale of values from “#000" to "#FFF".

1. 0:"#FFF
2. 1:"#EEE"
3. 2:"#DDD"
4. 3:"#CCC"
5. 4:"#BBB"
6. 5:"#AAA"
7. 6:"#999"
8. 7:"#888"
9. 8:"#777"
10. 9:"#666"
11. 10:"#555"
12. 11:"#444"
13. 12:"#333"
14. 13:"#222"
15. 14:"#111"
16. 15:"#000"

Later each div is generated based upon the required starting point from the user and is stacked upon each other which later is appended to a div which is on the DOM already.



**For Starting value=3 and height/Width 8px**



**For Starting value=10 and height/Width 25px**

**Finally, could you suggest and discuss ways to make this function and its output cooler and more flexible?**

To make the function cooler, we can use other css properties like border-radius. After varying border-radius we can manage to get many desired shapes.

We can write function having more number of color choices.

We can make a UI and provide properties to the user, so he is free to implement anything he likes.