In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

Generating the Symmetric Matrices

To generate symmetric matrix from any general matrix M, we can do any one of the following

M_symm = (M+M.T)/2

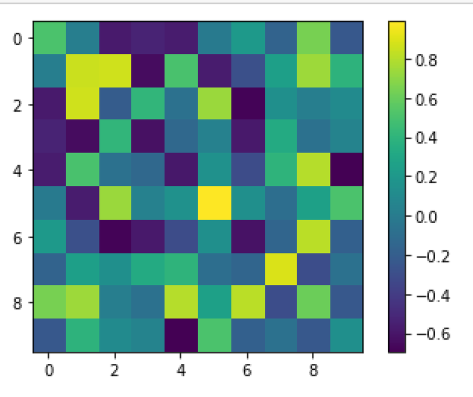
M_symm = M*M.T

Using rules of matrix transpose, it can be shown that these two matrices are symmetric for any matrix M

In [2]:
```python
N = 10
A = np.random.rand(N,N)
A_Symm = (A + A.T)/2
A_Symm*=2
A_Symm-=1
print(A_Symm)
```

```
[[ 0.51362694  0.02375238 -0.57354865 -0.52947567 -0.56655558 -0.0010004
3
   0.21929146 -0.1587675   0.64460971 -0.22849717]
 [ 0.02375238  0.85687269  0.86637343 -0.6393855   0.50356679 -0.5671016
1
  -0.28202469  0.25638067  0.74370076  0.39420788]
 [-0.57354865  0.86637343 -0.20362728  0.41619781 -0.0674289   0.7366903
3
  -0.6785701   0.14691901  0.02618289  0.10853407]
 [-0.52947567 -0.6393855   0.41619781 -0.61600671 -0.12818749  0.0486808
3
  -0.57794609  0.33673342 -0.06714558  0.05619415]
 [-0.56655558  0.50356679 -0.0674289  -0.12818749 -0.58481996  0.1593887
6
  -0.3046379   0.39828063  0.80487648 -0.69462498]
 [-0.00100043 -0.56710161  0.73669033  0.04868083  0.15938876  0.9906803
8
   0.14516898 -0.08241913  0.26028619  0.51602895]
 [ 0.21929146 -0.28202469 -0.6785701  -0.57794609 -0.3046379   0.1451689
8
  -0.61249417 -0.1417148   0.81758531 -0.17464582]
 [-0.1587675   0.25638067  0.14691901  0.33673342  0.39828063 -0.0824191
3
  -0.1417148   0.89679732 -0.2961392  -0.0670925 ]
 [ 0.64460971  0.74370076  0.02618289 -0.06714558  0.80487648  0.2602861
9
   0.81758531 -0.2961392   0.6153814  -0.23107845]
 [-0.22849717  0.39420788  0.10853407  0.05619415 -0.69462498  0.5160289
5
  -0.17464582 -0.0670925  -0.23107845  0.14642859]]
```

In [3]:
```python
plt.imshow(A_Symm)
plt.colorbar()
plt.savefig("Original Matrix")
plt.show()
```



Eigenvectors and Eigenvalues

There are two standard methods to compute the eigenvectors and eigenvalues, one for any general matrices, the second one is specifically for real symmetric matrices Since, this is a symmetric matrix, the eigenvalues are all always real the set of eigenvectors are orthogonal and hence represent the "dimensions" or "degrees of freedom" in which the data exists

In [4]:
```python
W, V = np.linalg.eig(A_Symm)
print("The eigenvalues are \n")
print(W)
print("The eigenvectors are \n")
print(V)
```

```
The eigenvalues are

[ 2.17746757  2.02819928  1.5618529  -2.14835398  0.9576927   0.54601796
 -1.73757565 -0.10570437 -0.88034648 -0.39641073]
The eigenvectors are

[[-0.40590252 -0.32845758  0.04840936  0.28806855 -0.08100624  0.5476180
5
  -0.00276859 -0.54917049 -0.16996785 -0.07140934]
 [-0.40134277  0.62980716 -0.09682022 -0.05486476 -0.37679816  0.0843203
9
  -0.49936468 -0.00916899  0.03708314  0.17312288]
 [ 0.1448123   0.45279265  0.24728691  0.30873667 -0.1055309  -0.0894596
6
   0.45153661 -0.40563666  0.47253523 -0.0817515 ]
 [ 0.29327398  0.07109931  0.01274624  0.2316118   0.16632393 -0.1054714
4
  -0.55627518 -0.16478143 -0.02845117 -0.69205233]
 [-0.16921227  0.26857131 -0.02574508  0.56672736  0.38956908 -0.2793262
8
   0.10563638  0.06342577 -0.53947778  0.20344179]
 [ 0.16620644  0.04174608  0.83356909 -0.14882385  0.22997872  0.1479000
7
  -0.27297487 -0.05236924 -0.08134798  0.30823168]
 [-0.26138429 -0.24660095  0.09901254  0.49551921  0.1267833   0.0604635
8
  -0.19828155  0.46213957  0.58032281  0.0782974 ]
 [ 0.16403861  0.35528614 -0.27823386 -0.10408752  0.53109801  0.6622390
6
   0.09112728  0.14084193  0.09758868 -0.01802605]
 [-0.62152251  0.11240367  0.28014482 -0.33982411  0.25601971 -0.1235677
6
   0.23392453  0.14087289  0.00256769 -0.50172314]
 [ 0.17149017  0.10878474  0.25645697  0.2232972  -0.49428862  0.3413125
7
   0.22200248  0.49660216 -0.31887032 -0.29108861]]
```

In [5]:
```python
## Get the dimensions of the matrices

print("A_Symm ", A_Symm.shape)
print("Eigenvectors ", V.shape)
print("Eigenvalues ", W.shape)
```

```
A_Symm  (10, 10)
Eigenvectors  (10, 10)
Eigenvalues  (10,)
```

In [6]:
```python
w, v = np.linalg.eigh(A_Symm)
print("The eigenvalues are \n")
print(w)
print("The eigenvectors are \n")
print(v)
```

```
The eigenvalues are

[-2.14835398 -1.73757565 -0.88034648 -0.39641073 -0.10570437  0.54601796
  0.9576927   1.5618529   2.02819928  2.17746757]
The eigenvectors are

[[ 0.28806855  0.00276859  0.16996785  0.07140934  0.54917049 -0.5476180
5
   0.08100624  0.04840936  0.32845758 -0.40590252]
 [-0.05486476  0.49936468 -0.03708314 -0.17312288  0.00916899 -0.0843203
9
   0.37679816 -0.09682022 -0.62980716 -0.40134277]
 [ 0.30873667 -0.45153661 -0.47253523  0.0817515   0.40563666  0.0894596
6
   0.1055309   0.24728691 -0.45279265  0.1448123 ]
 [ 0.2316118   0.55627518  0.02845117  0.69205233  0.16478143  0.1054714
4
  -0.16632393  0.01274624 -0.07109931  0.29327398]
 [ 0.56672736 -0.10563638  0.53947778 -0.20344179 -0.06342577  0.2793262
8
  -0.38956908 -0.02574508 -0.26857131 -0.16921227]
 [-0.14882385  0.27297487  0.08134798 -0.30823168  0.05236924 -0.1479000
7
  -0.22997872  0.83356909 -0.04174608  0.16620644]
 [ 0.49551921  0.19828155 -0.58032281 -0.0782974  -0.46213957 -0.0604635
8
  -0.1267833   0.09901254  0.24660095 -0.26138429]
 [-0.10408752 -0.09112728 -0.09758868  0.01802605 -0.14084193 -0.6622390
6
  -0.53109801 -0.27823386 -0.35528614  0.16403861]
 [-0.33982411 -0.23392453 -0.00256769  0.50172314 -0.14087289  0.1235677
6
  -0.25601971  0.28014482 -0.11240367 -0.62152251]
 [ 0.2232972  -0.22200248  0.31887032  0.29108861 -0.49660216 -0.3413125
7
   0.49428862  0.25645697 -0.10878474  0.17149017]]
```

Generating Approximation to the matrices

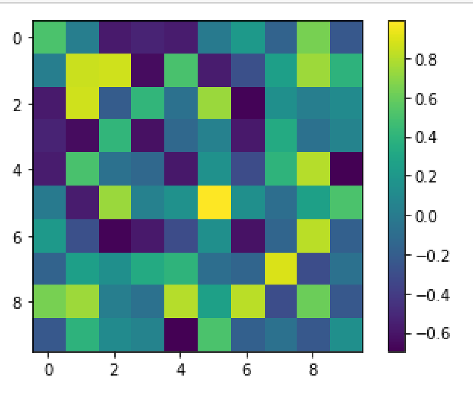
There are many methods to generate approximations to the matrices. We work with the following three methods and configurations

1. Eigenvalue Decomposition (2 eigenvectors)
2. Eigenvalue Decomposition (4 eigenvectors)

In [7]:
```python
## Before moving ahead, first we reconstruct the matrix
## The entire matrix through the entrie set of eignevalues and eigenvect
ors

Lambda = np.diag(W)
A_recon = V @ Lambda @ np.linalg.inv(V)
print(A_recon)
```

```
[[ 0.51362694  0.02375238 -0.57354865 -0.52947567 -0.56655558 -0.0010004
3
   0.21929146 -0.1587675   0.64460971 -0.22849717]
 [ 0.02375238  0.85687269  0.86637343 -0.6393855   0.50356679 -0.5671016
1
  -0.28202469  0.25638067  0.74370076  0.39420788]
 [-0.57354865  0.86637343 -0.20362728  0.41619781 -0.0674289   0.7366903
3
  -0.6785701   0.14691901  0.02618289  0.10853407]
 [-0.52947567 -0.6393855   0.41619781 -0.61600671 -0.12818749  0.0486808
3
  -0.57794609  0.33673342 -0.06714558  0.05619415]
 [-0.56655558  0.50356679 -0.0674289  -0.12818749 -0.58481996  0.1593887
6
  -0.3046379   0.39828063  0.80487648 -0.69462498]
 [-0.00100043 -0.56710161  0.73669033  0.04868083  0.15938876  0.9906803
8
   0.14516898 -0.08241913  0.26028619  0.51602895]
 [ 0.21929146 -0.28202469 -0.6785701  -0.57794609 -0.3046379   0.1451689
8
  -0.61249417 -0.1417148   0.81758531 -0.17464582]
 [-0.1587675   0.25638067  0.14691901  0.33673342  0.39828063 -0.0824191
3
  -0.1417148   0.89679732 -0.2961392  -0.0670925 ]
 [ 0.64460971  0.74370076  0.02618289 -0.06714558  0.80487648  0.2602861
9
   0.81758531 -0.2961392   0.6153814  -0.23107845]
 [-0.22849717  0.39420788  0.10853407  0.05619415 -0.69462498  0.5160289
5
  -0.17464582 -0.0670925  -0.23107845  0.14642859]]
```

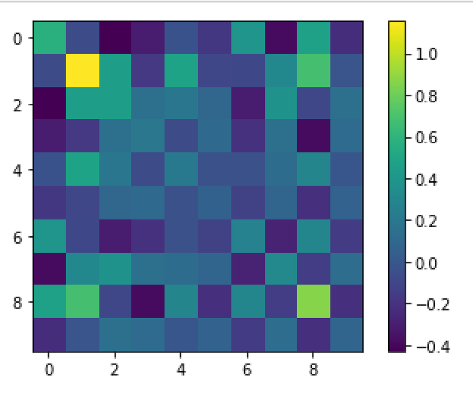
In [8]:
```python
plt.imshow(A_recon)
plt.colorbar()
plt.savefig("Reconstructed Matrix")
plt.show()
```



In [9]:
```python
## Generating the matrix of the top 2 eigenvalues and the matrices conta
ining the eiegenvectors
E_2 = np.zeros((2,2))
np.fill_diagonal(E_2,[W[0],W[1]])

Q_2 = V[:,0:2]
A_2 = np.dot(np.dot(Q_2,E_2),Q_2.T)
print(A_2)
```

```
[[ 0.57756374 -0.06484069 -0.42963109 -0.30657191 -0.02935964 -0.1747101
3
   0.39530177 -0.38166742  0.4744454  -0.22403974]
 [-0.06484069  1.15523737  0.45183271 -0.16547488  0.49094257 -0.0919242
4
  -0.08657518  0.31047832  0.68673685 -0.01090828]
 [-0.42963109  0.45183271  0.46148662  0.15777067  0.19328676  0.0907465
7
  -0.30888766  0.37800368 -0.09275469  0.15397786]
 [-0.30657191 -0.16547488  0.15777067  0.19753595 -0.0693291   0.1121584
7
  -0.20247933  0.1559877  -0.3806919   0.12519984]
 [-0.02935964  0.49094257  0.19328676 -0.0693291   0.20864211 -0.0384997
  -0.03801937  0.13308938  0.29023069 -0.00392947]
 [-0.17471013 -0.09192424  0.09074657  0.11215847 -0.0384997   0.0636862
5
  -0.11547691  0.08944893 -0.21541753  0.0712746 ]
 [ 0.39530177 -0.08657518 -0.30888766 -0.20247933 -0.03801937 -0.1154769
1
   0.2721073  -0.27106198  0.29752379 -0.15201395]
 [-0.38166742  0.31047832  0.37800368  0.1559877   0.13308938  0.0894489
3
  -0.27106198  0.31460878 -0.14100376  0.13964368]
 [ 0.4744454   0.68673685 -0.09275469 -0.3806919   0.29023069 -0.2154175
3
   0.29752379 -0.14100376  0.8667599  -0.20728496]
 [-0.22403974 -0.01090828  0.15397786  0.12519984 -0.00392947  0.0712746
  -0.15201395  0.13964368 -0.20728496  0.08803883]]
```

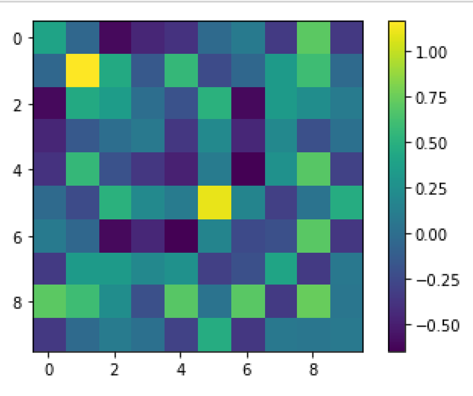
In [10]:
```python
plt.imshow(A_2)
plt.colorbar()
plt.savefig("2-Rank Approximation")
plt.show()
```



In [11]:
```python
## Generating the matrix of the top 2 eigenvalues and the matrices conta
ining the eiegenvectors
E_4 = np.zeros((4,4))
np.fill_diagonal(E_4,[W[0],W[1],W[2],W[3]])

Q_4 = V[ : ,0:4 ]
A_4 = np.dot(np.dot(Q_4,E_4),Q_4.T)
print(A_4)
```

```
[[ 0.40294598 -0.03820677 -0.602003   -0.44894653 -0.38203857 -0.0195822
9
   0.09612437 -0.33828723  0.70593472 -0.34284212]
 [-0.03820677  1.16341157  0.45082871 -0.14010251  0.56163526 -0.2355173
9
  -0.04314143  0.34028386  0.60431894 -0.02336962]
 [-0.602003    0.45082871  0.35221768  0.00907113 -0.19255312  0.5114036
8
  -0.5993123   0.33958131  0.24084153  0.10492055]
 [-0.44894653 -0.14010251  0.00907113  0.08254335 -0.35183616  0.2028053
4
  -0.4470707   0.20224099 -0.20602376  0.01919618]
 [-0.38203857  0.56163526 -0.19255312 -0.35183616 -0.4803308   0.1091801
1
  -0.64531066  0.27100695  0.69271244 -0.28611288]
 [-0.01958229 -0.23551739  0.51140368  0.20280534  0.10918011  1.1013372
   0.17185925 -0.30606668  0.04065575  0.47655307]
 [ 0.09612437 -0.04314143 -0.5993123  -0.4470707  -0.64531066  0.1718592
5
  -0.24008641 -0.20328246  0.70260621 -0.35006586]
 [-0.33828723  0.34028386  0.33958131  0.20224099  0.27100695 -0.3060666
8
  -0.20328246  0.41224246 -0.33873399  0.07813066]
 [ 0.70593472  0.60431894  0.24084153 -0.20602376  0.69271244  0.0406557
5
   0.70260621 -0.33873399  0.74124303  0.06794742]
 [-0.34284212 -0.02336962  0.10492055  0.01919618 -0.28611288  0.4765530
7
  -0.35006586  0.07813066  0.06794742  0.08364172]]
```

In [12]:
```python
plt.imshow(A_4)
plt.colorbar()
plt.savefig("4-rank Approximation")
plt.show()
```



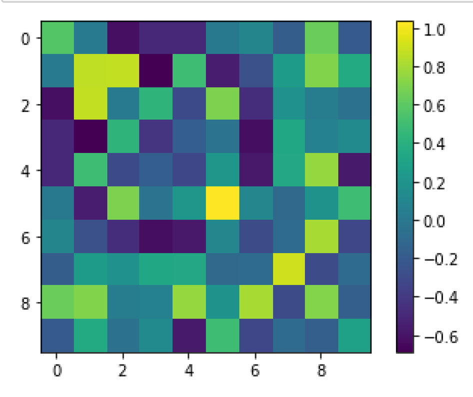
In [13]:
```python
## Generating the matrix of the top 2 eigenvalues and the matrices conta
ining the eiegenvectors
E_7 = np.zeros((7,7))
np.fill_diagonal(E_7,W[:7])

Q_7 = V[ : ,0:7 ]
A_7 = np.dot(np.dot(Q_7,E_7),Q_7.T)
print(A_7)
```

```
[[ 0.57295994  0.0138352  -0.6183931  -0.49606276 -0.49527396  0.0054864
8
   0.10341395 -0.18103532  0.65025037 -0.20137206]
 [ 0.0138352   0.86997323  0.87658254 -0.68764862  0.49985527 -0.5485532
9
  -0.25815399  0.25819296  0.70921591  0.36333997]
 [-0.6183931   0.87658254  0.01298693  0.43385516 -0.30116133  0.6951065
5
  -0.4595117   0.1820606   0.03747017 -0.03597403]
 [-0.49606276 -0.68764862  0.43385516 -0.42256837 -0.17159161 -0.0329288
2
```

```
  -0.62201084  0.33678114  0.06797758  0.13538735]
 [-0.49527396  0.49985527 -0.30116133 -0.17159161 -0.3117751   0.2225298
5
  -0.57083631  0.35142362  0.76413926 -0.56333061]
 [ 0.00548648 -0.54855329  0.69510655 -0.03292882  0.22252985  1.0344576
7
   0.11061813 -0.09239007  0.19801877  0.50054858]
 [ 0.10341395 -0.25815399 -0.4595117  -0.62201084 -0.57083631  0.1106181
3
  -0.29101008 -0.08553753  0.81020632 -0.32232761]
 [-0.18103532  0.25819296  0.1820606   0.33678114  0.35142362 -0.0923900
7
  -0.08553753  0.90740695 -0.29023617 -0.08501399]
 [ 0.65025037  0.70921591  0.03747017  0.06797758  0.76413926  0.1980187
7
   0.81020632 -0.29023617  0.71727185 -0.16651024]
 [-0.20137206  0.36333997 -0.03597403  0.13538735 -0.56333061  0.5005485
8
  -0.32232761 -0.08501399 -0.16651024  0.29559776]]
```

In [14]:

```
plt.imshow(A_7)
plt.colorbar()
plt.savefig("7-rank Approximation")
plt.show()
```



Observations and Comments

The above method works only for square matrices, for general rectnagular marices we need to perform Singular Value Decomposition (SVD). To measure the variance of data represented by the approximation, we can measure the frobenius norm of the difference of the approximation matrix and the original matrix. The approximated matrix essentially reduces the complexity in terms of dimensionality of the original matrix, as instead of 10 eigenvectors and values, we using the 'top 2' or 'top 4' vectors according to their 'importance'