In [170]:
```
#Submitted by : Philani Mpofu (1848751)
#Submitted by : Matthew Kruger (1669326)
#Submitted by : Chloe Smith (1877342)
#Submitted by : Jesse Bristow (1875955)
```

In [171]:
```
import numpy as np
from numpy import linalg
import pandas as pd
import matplotlib.pyplot as plt
```

In [172]:
```
# Load data
dataset = datasets.fetch_olivetti_faces()
images = dataset['data']
target = dataset['target']
print('Images Shape: ', images.shape)
print(images)

#print(target)
#print(images)
```

```
Images Shape:  (400, 4096)
[[0.30991736 0.3677686  0.41735536 ... 0.15289256 0.16115703 0.1570248
]
 [0.45454547 0.47107437 0.5123967  ... 0.15289256 0.15289256 0.1528925
6]
 [0.3181818  0.40082645 0.49173555 ... 0.14049587 0.14876033 0.1528925
6]
 ...
 [0.5        0.53305787 0.607438   ... 0.17768595 0.14876033 0.1900826
4]
 [0.21487603 0.21900827 0.21900827 ... 0.57438016 0.59090906 0.6033057
6]
 [0.5165289  0.46280992 0.28099173 ... 0.35950413 0.3553719  0.3842975
2]]
```

In [173]:
```
# Normalize the data, then get the Covariance Matrix
# targetNew = target.reshape(400,1) # Need to rearrange target vector i
n order to make a matrix to find covariant matrix

covMatrix = (1/images.shape[0])*(np.dot(images.T,images))
print('Covariance Matrix:\n', covMatrix)
print('Covariance Matrix Shape:',covMatrix.shape)
```

```
Covariance Matrix:
 [[0.19267644 0.2059697  0.21917908 ... 0.12233528 0.11950444 0.1195607
7]
 [0.2059697  0.22438237 0.24084835 ... 0.1306454  0.12774651 0.1280675
1]
 [0.21917908 0.24084835 0.26467338 ... 0.14092542 0.13807441 0.1383975
9]
 ...
 [0.12233528 0.1306454  0.14092542 ... 0.13850415 0.13217118 0.1277516
2]
 [0.11950444 0.12774651 0.13807441 ... 0.13217118 0.13200496 0.1286022
2]
 [0.11956077 0.12806751 0.13839759 ... 0.12775162 0.12860222 0.1289292
9]]
Covariance Matrix Shape: (4096, 4096)
```

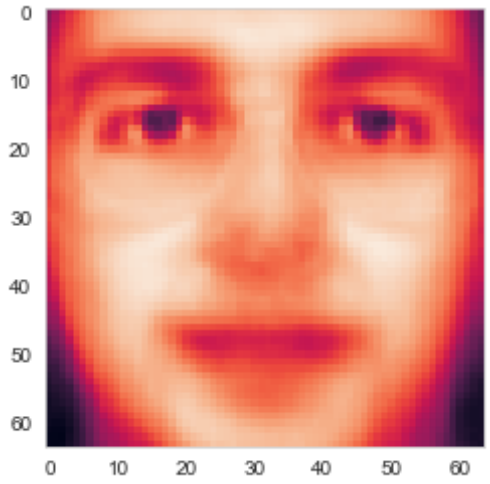
In [174]:
```
# Compute eigenvalues/vectors and convert values to their real number r
epresentations
eigvalues, eigvectors = np.linalg.eig(covMatrix)
eigvalues = eigvalues.real
eigvectors = eigvectors.real
```

In [187]:
```
print('Eigenvalues Shape:',eigvalues.shape)
print('Eigenvectors Shape:',eigvectors.shape)

# # Make a list of (eigenvalue, eigenvector) tuples
# eig_pairs = [(np.abs(eigvalues[i]), eigvectors[:,i]) for i in range(l
en(eigvalues))]
# # Sort the (eigenvalue, eigenvector) tuples from high to low
# eig_pairs.sort(key=lambda x: x[0], reverse=True)
```

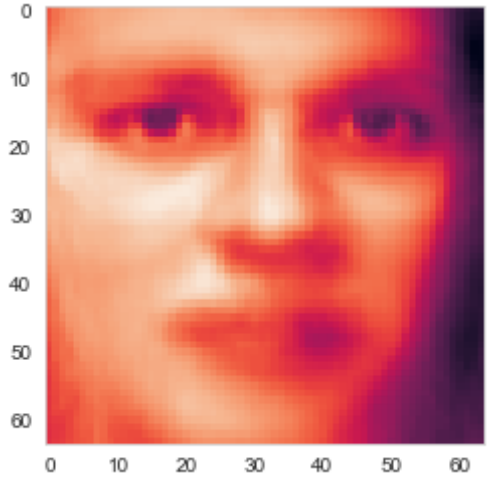
```
eigvalues_sum = np.sum(eigvalues) # Sum up all of the eigenvalues

# Use variation matrix to store the variation of each eigenvalue
var_matrix = ([])
for i in range(len(eigvalues)):
    var_matrix = np.append(var_matrix,(eigvalues[i]/eigvalues_sum))

X = eigvectors[:5] # Store the eigenvectors with the 5 lowest variation
s.
projection = np.dot(images,X.T)
print('Projection Matrix: \n', projection)
print('Projection Matrix Shape: ', projection.shape)
```

```
Eigenvalues Shape: (4096,)
Eigenvectors Shape: (4096, 4096)

Projection Matrix:
 [[ 1.0470986  -0.17621446  0.02594492 -0.1250127   0.1380943 ]
 [ 0.88770133 -0.3319137  -0.05387758 -0.25483778  0.3723411 ]
 [ 0.8913089  -0.25974685 -0.10215159 -0.21585543  0.11841822]
 ...
 [ 0.6764893  -0.5245818  -0.11842033 -0.12979957  0.3205065 ]
 [ 1.0825269  -0.5509006  -0.12855414 -0.14033079  0.35880226]
 [ 1.0696241  -0.45040724 -0.01769611  0.01406448  0.4974962 ]]
Projection Matrix Shape:  (400, 5)
```

In [189]:
```
projection = np.dot(images,eigvectors[:,:5])
image = np.sum(projection[0]*eigvectors[:,:5],axis=1)
plt.imshow(image.reshape((64,64)))
plt.show()
```

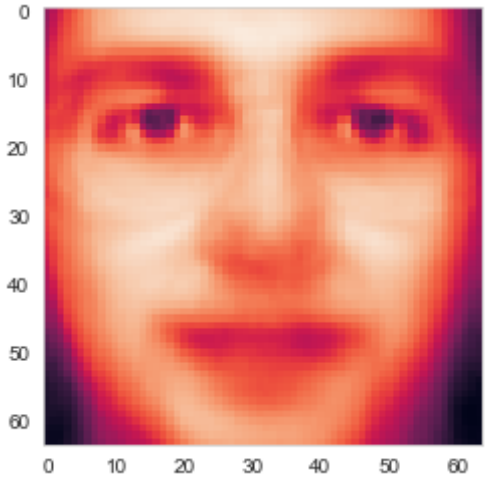


In [190]:
```
projection = np.dot(images,eigvectors[:,:5])
image = np.sum(projection[1]*eigvectors[:,:5],axis=1)
plt.imshow(image.reshape((64,64)))
plt.show()
```

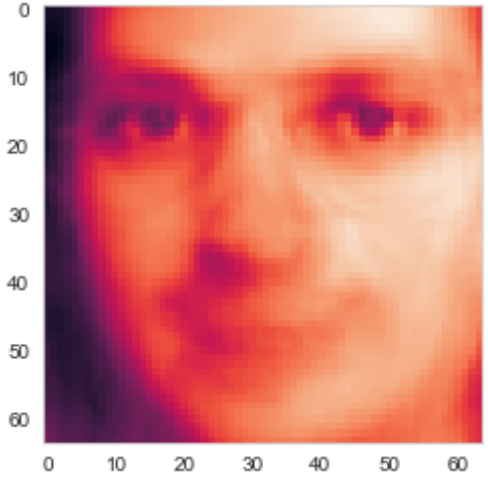


In [191]:
```
projection = np.dot(images,eigvectors[:,:5])
image = np.sum(projection[2]*eigvectors[:,:5],axis=1)
```

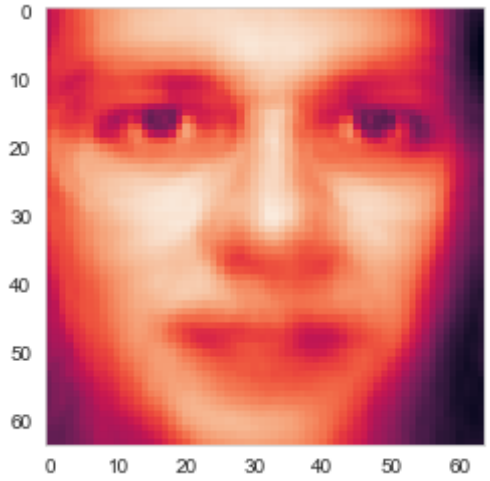
```
plt.imshow(image.reshape((64,64)))
plt.show()
```



In [192]:
```
projection = np.dot(images,eigvectors[:,:5])
image = np.sum(projection[3]*eigvectors[:,:5],axis=1)
plt.imshow(image.reshape((64,64)))
plt.show()
```



In [193]:
```
projection = np.dot(images,eigvectors[:,:5])
image = np.sum(projection[4]*eigvectors[:,:5],axis=1)
plt.imshow(image.reshape((64,64)))
plt.show()
```



In [ ]:
```

```