Karanjot Singh

In [46]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [47]:

```
def sigmoid(x):
    return 1/(1+np.exp(-x))

def sigmoid_deriv(x):
    return sigmoid(x)*(1-sigmoid(x))

def forword(x,w1,w2,predict = False):
    a1 = np.matmul(x,w1)
    z1 = sigmoid(a1)

    bias = np.ones((len(z1),1))
    z1 = np.concatenate((bias,z1),axis = 1)
    a2 = np.matmul(z1,w2)
    z2 = sigmoid(a2)
    if predict:
        return z2
    return a1,z1,a2,z2

def backprop(a2,z0,z1,z2,y):
    delta2 = z2-y
    Delta2 = np.matmul(z1.T,delta2)
    delta1 = (delta2.dot(w2[1:,:].T))*sigmoid_deriv(a1)
    Delta1 = np.matmul(z0.T,delta1)
    return delta2,Delta1,Delta2
```

In [48]:

```
x = np.array([[1,1,0],[1,0,1],[1,0,0],[1,1,1]])
y = np.array([[1],[1],[0],[0]])
```

In [49]:

```
w1 = np.random.randn(3,5)
w2 = np.random.randn(6,1)
```

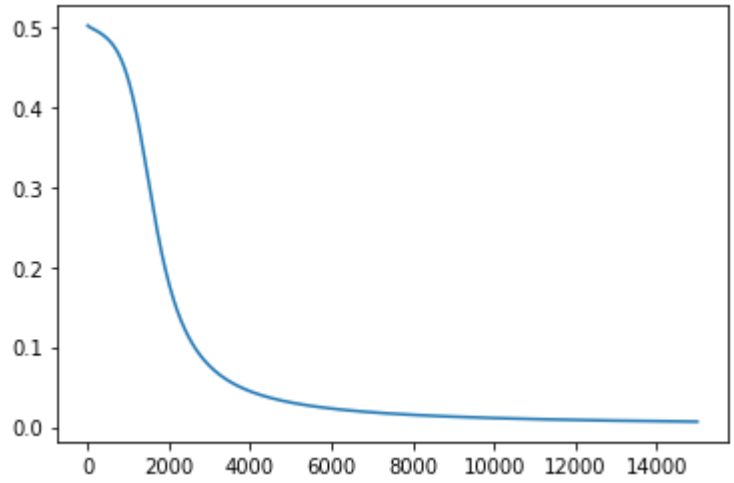
In [50]:

```
lr = 0.09
costs = []
epochs = 15000
m = len(x)
for i in range(epochs):
    a1,z1,a2,z2 = forword(x,w1,w2)
    delta2,Delta1,Delta2 = backprop(a2,x,z1,z2,y)
    w1-=lr*(1/m)*Delta1
    w2-=lr*(1/m)*Delta2

    c = np.mean(np.abs(delta2))
    costs.append(c)

    if i% 1000 == 0:
        print(f"Iteration: {i}, Error:{c}")

print("Training Complete.")

z3 = forword(x,w1,w2,True)
print("Percentages: ")
print(z3)
print("Predictions:  ")
print(np.round(z3))
```

```
Iteration: 0, Error:0.5018898760246621
Iteration: 1000, Error:0.433955039341291
Iteration: 2000, Error:0.17906240393036757
Iteration: 3000, Error:0.07619599338380037
Iteration: 4000, Error:0.044478371830694544
Iteration: 5000, Error:0.030547708941775067
Iteration: 6000, Error:0.022962730523866236
Iteration: 7000, Error:0.018260839024719054
Iteration: 8000, Error:0.015086834323718691
Iteration: 9000, Error:0.012811927799152169
Iteration: 10000, Error:0.01110758235380579
Iteration: 11000, Error:0.009786495334630966
Iteration: 12000, Error:0.008734520929789446
Iteration: 13000, Error:0.007878340200354144
Iteration: 14000, Error:0.007168812216110311
Training Complete.
Percentages:
[[0.99282905]
 [0.99351859]
 [0.00309113]
 [0.0095438 ]]
Predictions:
[[1.]
 [1.]
 [0.]
 [0.]]
```

In [51]:

```
plt.plot(costs)
plt.show()
```



In [ ]:

```

```

In [ ]:

```

```