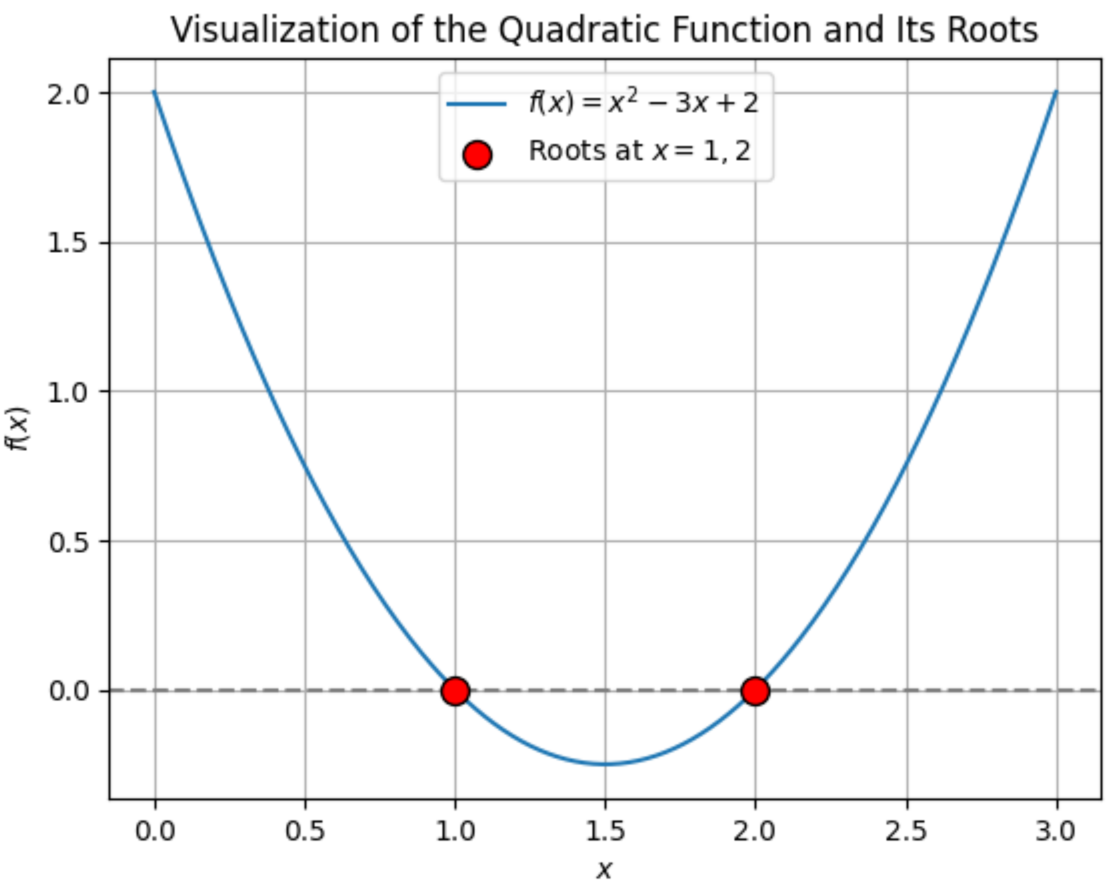
In [14]:
```
import matplotlib.pyplot as plt
import numpy as np

# Define the quadratic function
def quadratic_equation(x):
    return x ** 2 - 3 * x + 2

# Generate x values
x_values = np.linspace(0, 3, 301)

# Plot the function
plt.plot(x_values, quadratic_equation(x_values), label=r"$f(x) = x^2 - 3x +
plt.axhline(0, color='gray', linestyle='--')  # x-axis

# # Highlight the roots with circles
roots = [1, 2]
plt.scatter(roots, [0, 0], color='red', edgecolor='black', s=100, zorder=5,

# # Format and display the plot
plt.xlabel("$x$")
plt.ylabel("$f(x)$")
plt.title("Visualization of the Quadratic Function and Its Roots")
plt.legend()
plt.grid(True)
plt.show()
```



In [15]:
```python
from scipy.optimize import minimize

# Define the objective function, which is the absolute value of the quadrati
def objective_function(x):
    return abs(x ** 2 - 3 * x + 2)

# Perform the minimization starting from an initial guess of x = 0
result = minimize(
    fun=objective_function,  # Objective function to minimize
    x0=0,                    # Initial guess
    method="Nelder-Mead",    # Optimization method
    tol=1e-6                 # Tolerance for convergence
)

print(result)
```

```
       message: Optimization terminated successfully.
       success: True
        status: 0
           fun: 8.881784197001252e-16
             x: [ 1.000e+00]
           nit: 31
          nfev: 62
 final_simplex: (array([[ 1.000e+00],
                       [ 1.000e+00]]), array([ 8.882e-16,  9.766e-07]))
```

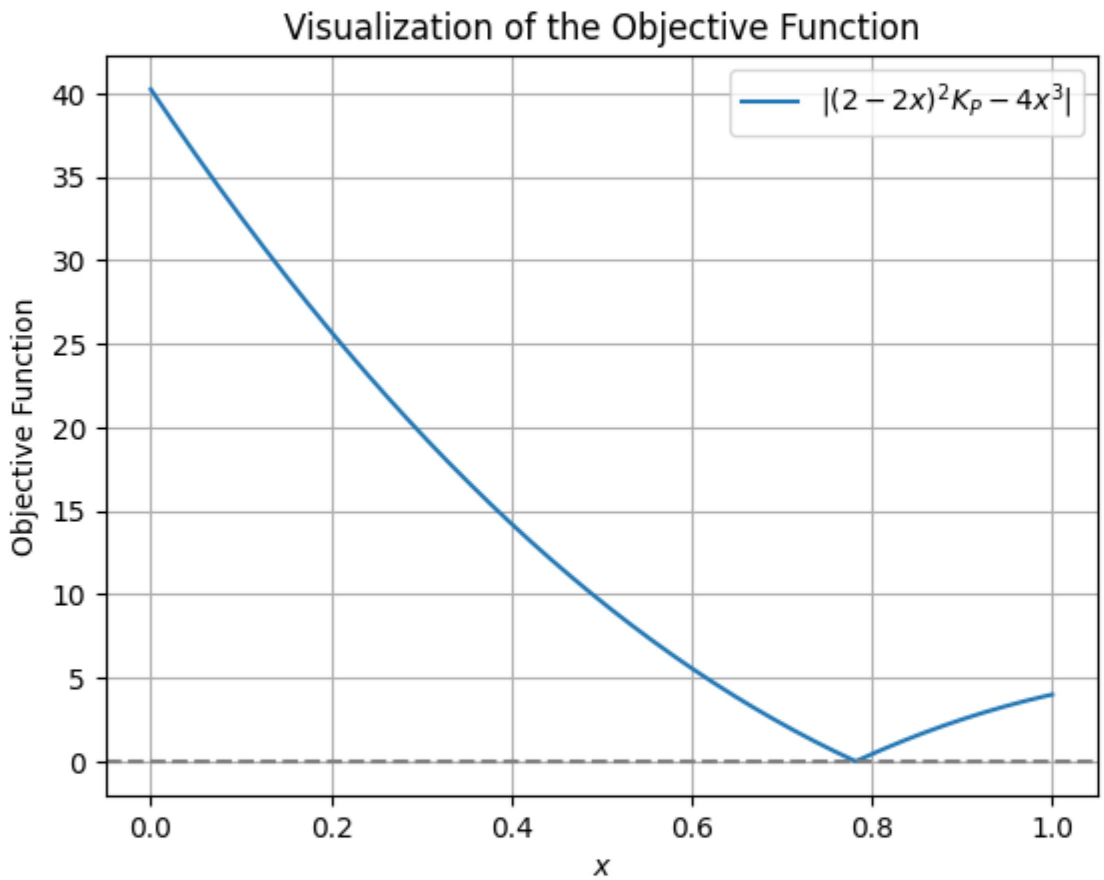
In [16]:
```python
# Perform the minimization starting from an initial guess of x = 0
result = minimize(
    fun=objective_function,  # Objective function to minimize
    x0=2.1,                    # Initial guess
    method="Nelder-Mead",    # Optimization method
    tol=1e-6                 # Tolerance for convergence
)

print(result)
```

```
       message: Optimization terminated successfully.
       success: True
        status: 0
           fun: 3.814698725790322e-07
             x: [ 2.000e+00]
           nit: 19
          nfev: 38
 final_simplex: (array([[ 2.000e+00],
                       [ 2.000e+00]]), array([ 3.815e-07,  4.196e-07]))
```

In [17]:
```python
def objective_function(x, K_P):
    equilibrium_equation = (2 - 2 * x) ** 2 * K_P - 4 * x ** 3
    return abs(equilibrium_equation)
```

In [18]:
```python
# Generate x values
x_values = np.linspace(0, 1, 400)

# Plot the function
plt.plot(x_values, objective_function(x_values, 10.060), label=r"$|(2 - 2x)^
plt.axhline(0, color='gray', linestyle='--')  # x-axis
```

```python
# Format and display the plot
plt.xlabel("$x$")
plt.ylabel("Objective Function")
plt.title("Visualization of the Objective Function")
plt.legend()
plt.grid(True)
plt.show()
```



In [19]:
```python
# Perform the minimization with an initial guess of x = 0
result = minimize(
    fun=objective_function,
    x0=0,
    args=(10.060,),
    method="Nelder-Mead",
    tol=1e-6
)

print("{:.0f}%".format(result["x"][0] * 100))  # Convert the result to perce
```

78%

In [ ]: