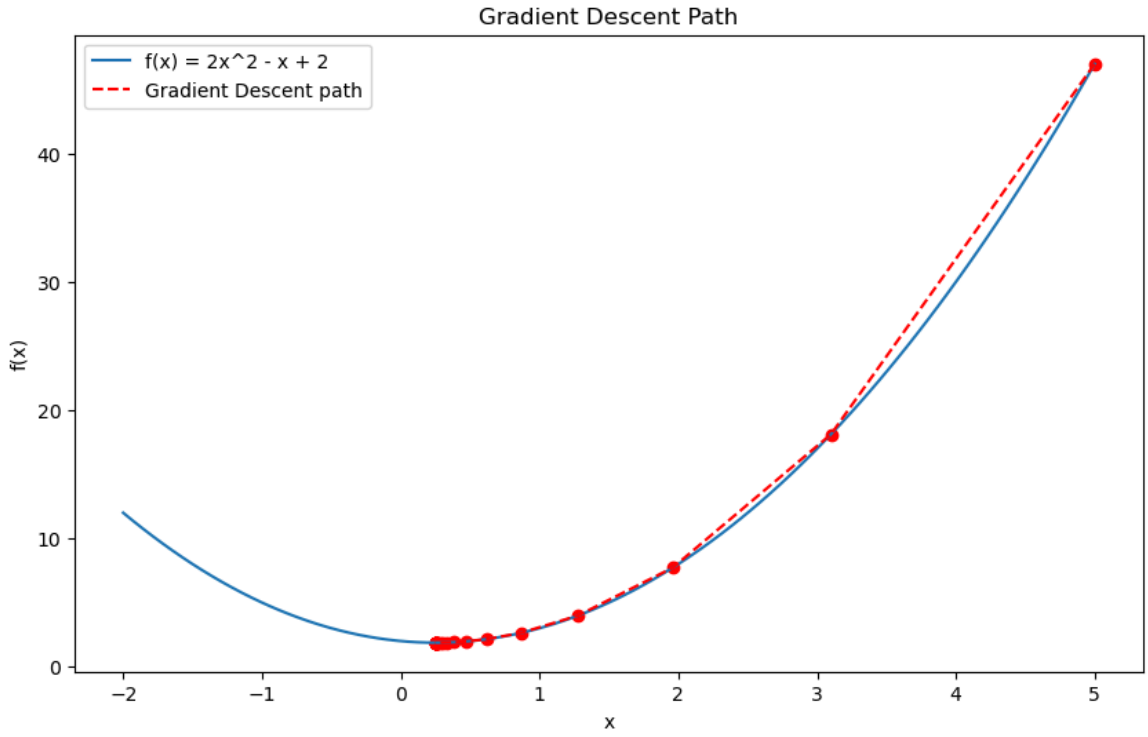
Name: Ameya Satish Khond
SJSU ID: 019136845

In [55]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

# 1) Define the function and its derivative

def f(x):

    return 2*x**2 - x + 2

def fprime(x):

    return 4*x - 1   # derivative of 2x^2 - x + 2
```

In [56]:
```
# Set up your own starting point, learning rate and epochs

x = 5
lr = 0.1
epochs = 25

x_values = [x]

for i in range(epochs):
    grad = fprime(x)
    x = x - lr * grad
    x_values.append(x)
    print(f"Epoch {i+1}: x = {x:.4f}, f(x) = {f(x):.4f}")

print(f"\nFinal x after gradient descent: {x:.4f}")
print(f"\nMinimum value of function: {f(x):.4f}")
```

```
Epoch 1: x = 3.1000, f(x) = 18.1200
Epoch 2: x = 1.9600, f(x) = 7.7232
Epoch 3: x = 1.2760, f(x) = 3.9804
Epoch 4: x = 0.8656, f(x) = 2.6329
Epoch 5: x = 0.6194, f(x) = 2.1479
Epoch 6: x = 0.4716, f(x) = 1.9732
Epoch 7: x = 0.3830, f(x) = 1.9104
Epoch 8: x = 0.3298, f(x) = 1.8877
Epoch 9: x = 0.2979, f(x) = 1.8796
Epoch 10: x = 0.2787, f(x) = 1.8766
Epoch 11: x = 0.2672, f(x) = 1.8756
Epoch 12: x = 0.2603, f(x) = 1.8752
Epoch 13: x = 0.2562, f(x) = 1.8751
Epoch 14: x = 0.2537, f(x) = 1.8750
Epoch 15: x = 0.2522, f(x) = 1.8750
Epoch 16: x = 0.2513, f(x) = 1.8750
Epoch 17: x = 0.2508, f(x) = 1.8750
Epoch 18: x = 0.2505, f(x) = 1.8750
Epoch 19: x = 0.2503, f(x) = 1.8750
Epoch 20: x = 0.2502, f(x) = 1.8750
Epoch 21: x = 0.2501, f(x) = 1.8750
Epoch 22: x = 0.2501, f(x) = 1.8750
Epoch 23: x = 0.2500, f(x) = 1.8750
Epoch 24: x = 0.2500, f(x) = 1.8750
Epoch 25: x = 0.2500, f(x) = 1.8750

Final x after gradient descent: 0.2500

Minimum value of function: 1.8750
```

In [57]:
```
# 2) Visualization of Gradient Descent

x_range = np.linspace(-2, 5, 100)
y_range = f(x_range)

plt.figure(figsize=(10, 6))
plt.plot(x_range, y_range, label="f(x) = 2x^2 - x + 2")
plt.scatter(x_values, [f(xi) for xi in x_values], color='red')
plt.plot(x_values, [f(xi) for xi in x_values], color='red', linestyle=
plt.title("Gradient Descent Path")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.show()
```



In [58]:
```
# 3) Using scipy.optimize to get similar result

result = optimize.minimize(f, x0 = 5, method='BFGS')
print("\nUsing scipy.optimize:")
print("\nMinimum at x =", result.x[0])
print("Function value =", result.fun)
```

```
Using scipy.optimize:

Minimum at x = 0.24999999374054904
Function value = 1.875
```

In [ ]: