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


n_rounds = 1000
n_arms = 3
success_probabilities = [0.6, 0.4, 0.7]

class UCB:
    def __init__(self, n_arms):
        self.n_arms = n_arms
        self.counts = np.zeros(n_arms)
        self.values = np.zeros(n_arms)
        self.total_reward = 0

    def select_arm(self):
        total_counts = np.sum(self.counts)
        if total_counts < self.n_arms:
            return int(total_counts)
        ucb_values = self.values + np.sqrt(2 * np.log(total_counts) / self.counts)
        return np.argmax(ucb_values)

    def update(self, chosen_arm, reward):
        self.counts[chosen_arm] += 1
        self.total_reward += reward
        self.values[chosen_arm] += (reward - self.values[chosen_arm]) / self.counts[chosen_arm]


class ThompsonSampling:
    def __init__(self, n_arms):
        self.n_arms = n_arms
        self.successes = np.zeros(n_arms)
        self.failures = np.zeros(n_arms)

    def select_arm(self):
        samples = [np.random.beta(self.successes[i] + 1, self.failures[i] + 1) for i in range(self.n_arms)]
        return np.argmax(samples)

    def update(self, chosen_arm, reward):
        if reward == 1:
            self.successes[chosen_arm] += 1
        else:
            self.failures[chosen_arm] += 1


def simulate(algorithm):
    rewards = np.zeros(n_rounds)
    arm_counts = np.zeros((n_rounds, n_arms))

    for round in range(n_rounds):
        chosen_arm = algorithm.select_arm()
        reward = 1 if np.random.rand() < success_probabilities[chosen_arm] else 0
        algorithm.update(chosen_arm, reward)

        rewards[round] = reward
        arm_counts[round] = algorithm.counts if isinstance(algorithm, UCB) else algorithm.successes + algorithm.failures

    return rewards, arm_counts


ucb = UCB(n_arms)
ucb_rewards, ucb_arm_counts = simulate(ucb)

ts = ThompsonSampling(n_arms)
ts_rewards, ts_arm_counts = simulate(ts)

plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)
plt.plot(np.cumsum(ucb_rewards), label='UCB')
plt.plot(np.cumsum(ts_rewards), label='Thompson Sampling')
plt.title('Cumulative Rewards Over Time')
plt.xlabel('Rounds')
plt.ylabel('Cumulative Reward')
plt.legend()
```



```
plt.subplot(1, 2, 2)
plt.plot(np.cumsum(ucb_arm_counts, axis=0), label=['Video Lectures', 'Interactive Quizzes', 'Gamified Modules'])
plt.title('Arm Selection Counts (UCB)')
```

```
plt.xlabel('Rounds')
plt.ylabel('Counts')
plt.legend()

plt.tight_layout()
plt.show()
```