In [2]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import random

# Define room and furniture parameters
ROOM_SIZE = (10, 10)  # 10x10 grid
FURNITURE_ITEMS = ['Bed', 'Table', 'Chair', 'Sofa']
NUM_SAMPLES = 500  # Dataset size

# Generate synthetic dataset
def generate_synthetic_data(num_samples):
    X = []
    Y = []
    for _ in range(num_samples):
        # Random room constraints
        room_width = random.randint(6, 10)
        room_height = random.randint(6, 10)

        # Generate furniture placement as output
        layout = np.zeros((room_width, room_height))
        furniture_positions = []
        for item in FURNITURE_ITEMS:
            x, y = random.randint(0, room_width - 1), random.randint(0, roo
            layout[x, y] = FURNITURE_ITEMS.index(item) + 1
            furniture_positions.append((x, y))

        X.append([room_width, room_height])
        Y.append(furniture_positions)

    return np.array(X), np.array(Y)

# Generate data
X_train, Y_train = generate_synthetic_data(NUM_SAMPLES)

# Define AI model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(2,)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(len(FURNITURE_ITEMS) * 2, activation='sigmoid')
])

model.compile(optimizer='adam', loss='mse')

# Reshape Y_train for training
Y_train_reshaped = Y_train.reshape(NUM_SAMPLES, -1) / max(ROOM_SIZE)  # Nor

# Train the model
model.fit(X_train, Y_train_reshaped, epochs=50, batch_size=32, verbose=1)

# Predict function
def generate_layout(room_size):
    input_data = np.array([room_size])
    prediction = model.predict(input_data)[0] * max(ROOM_SIZE)
    prediction = prediction.reshape(len(FURNITURE_ITEMS), 2).astype(int)

    # Generate a blank room grid
    room_grid = np.zeros(room_size)

    for i, (x, y) in enumerate(prediction):
        if 0 <= x < room_size[0] and 0 <= y < room_size[1]:
```

```
            room_grid[x, y] = i + 1

    return room_grid, prediction

# Visualization function
def plot_layout(room_size, furniture_positions):
    plt.figure(figsize=(6, 6))
    plt.xlim(0, room_size[0])
    plt.ylim(0, room_size[1])

    for i, (x, y) in enumerate(furniture_positions):
        plt.scatter(x, y, marker='s', s=500, label=FURNITURE_ITEMS[i])

    plt.legend()
    plt.grid()
    plt.title("Optimized Furniture Layout")
    plt.show()

# Generate and visualize a sample layout
room_size = (8, 8)
layout, positions = generate_layout(room_size)
plot_layout(room_size, positions)
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0576
Epoch 49/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0577
Epoch 50/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0576
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 208ms/step
```

In [ ]: