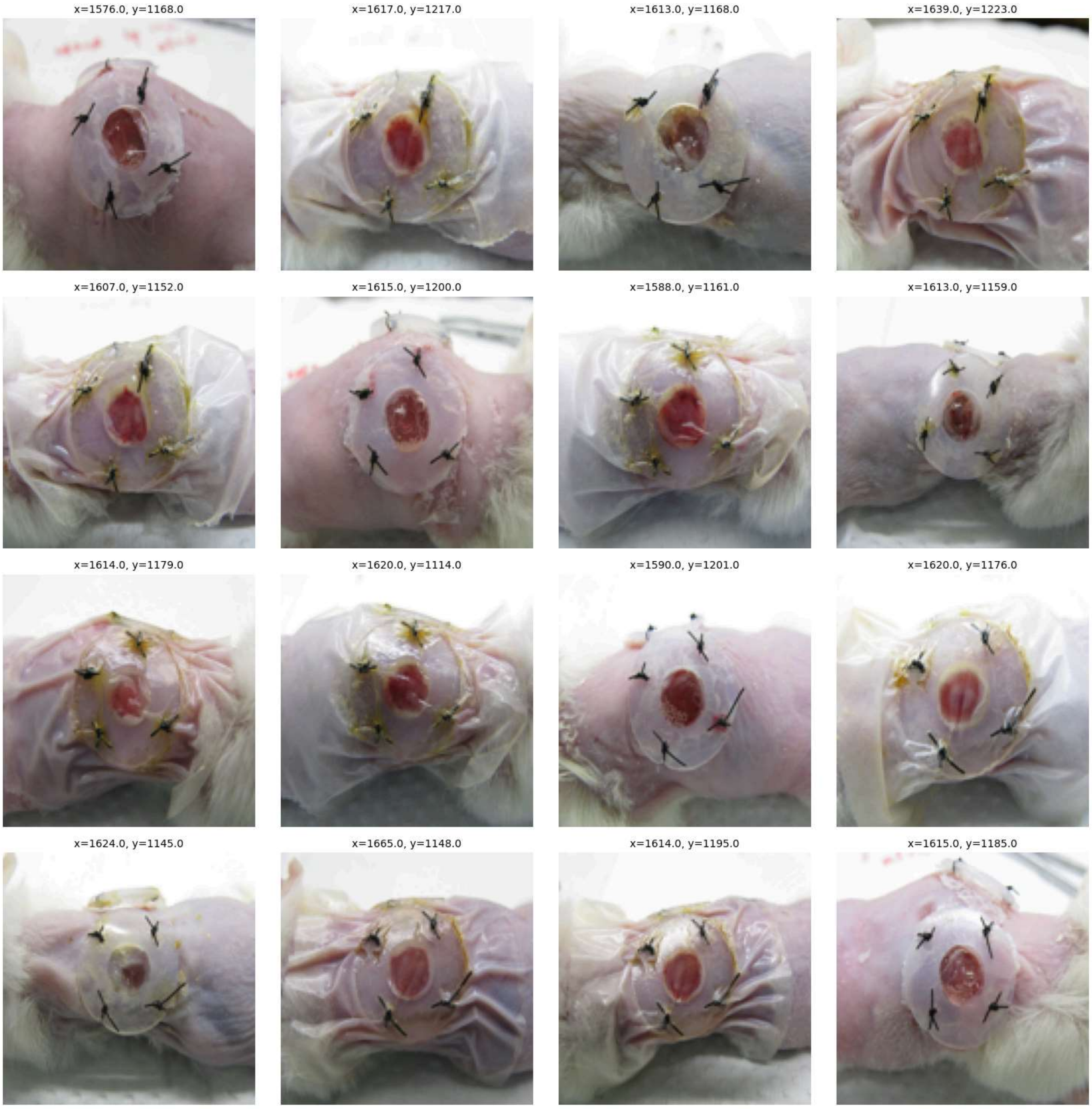
Simple Linear Regression Model

In [1]:
```
from Utils import *
```

In [2]:
```
# load training data
trainImages, trainOutputxi, trainOutputyi, trainOutputxw, trainOutputyw = readIm
```

In [3]:
```
# normalizing pixel values
trainImagesNormalized = np.array(trainImages) / 255.0
```

In [4]:
```
# show some training images
titles = [f"x={x}, y={y}" for x, y in zip(trainOutputxi[:16], trainOutputyi[:16]
display_images(trainImages[:16], rows=4, cols=4, titles=titles)
```



In [5]:
```
# Flatten the images for the linear regression model
trainImages_flattened =\
[img.flatten() for img in trainImagesNormalized]
trainImages_flattened = np.array(trainImages_flattened)
```

In [6]:
```
# Prepare the Outputs
trainOutputs = np.array([
    trainOutputxi,  # x values
    trainOutputyi,  # y values
    trainOutputxw,  # x_width values
    trainOutputyw   # y_width values
], dtype=float).T  # Transpose to align correctly
```

In [7]:
```
# Split the Data
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    trainImages_flattened, trainOutputs, test_size=0.2, random_state=42
)
```

In [8]:
```
# Train the Linear Regression Model
from sklearn.linear_model import LinearRegression

# Initialize the Linear Regression model
lr_model = LinearRegression()

# Train the model
lr_model.fit(X_train, y_train)
```

Out[8]:
```
▾ LinearRegression
LinearRegression()
```

In [9]:
```
# Evaluate the Model
from sklearn.metrics import mean_squared_error

# Predict on the validation set
y_pred = lr_model.predict(X_val)

# Calculate MSE for validation
mse = mean_squared_error(y_val, y_pred)
print(f"Mean Squared Error on Validation Set: {mse:.2f}")
```

```
Mean Squared Error on Validation Set: 1030.05
```

In [10]:
```
# Save the Model
import joblib

# Save the trained model to a file
joblib.dump(lr_model, "linear_regression_models/linear_regression_simple_model.p
```

Out[10]:
```
['linear_regression_models/linear_regression_simple_model.pkl']
```

Improving the model

In [11]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
```

In [12]:
```python
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Save the scaler
joblib.dump(scaler, "linear_regression_scalers/scaler.pkl")
```

Out[12]:
```
['linear_regression_scalers/scaler.pkl']
```

In [13]:
```python
# Define the Ridge regression model
ridge_model = Ridge()

# Define the hyperparameter grid for alpha
param_grid = {'alpha': [0.01, 0.1, 1, 10, 100, 1000]}  # A range of values for a
```

In [14]:
```python
# Set up GridSearchCV to find the best alpha
grid_search = GridSearchCV(ridge_model, param_grid, cv=5, scoring='neg_mean_squa
```

In [15]:
```python
# Fit the grid search on the training data
grid_search.fit(X_train_scaled, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
```

Out[15]:

▸ GridSearchCV

▸ best_estimator_: Ridge

▸ Ridge

In [16]:
```python
import joblib

# Save models for each combination of hyperparameters
for idx, alpha_value in enumerate(grid_search.cv_results_['param_alpha']):
    model = Ridge(alpha=alpha_value)
    model.fit(X_train_scaled, y_train)

    model_filename = f"linear_regression_models/ridge_model_alpha_{alpha_value}.
    joblib.dump(model, model_filename)
    print(f"Model with alpha {alpha_value} saved as {model_filename}")
```

```
Model with alpha 0.01 saved as linear_regression_models/ridge_model_alpha_0.01.pk
l
Model with alpha 0.1 saved as linear_regression_models/ridge_model_alpha_0.1.pkl
Model with alpha 1.0 saved as linear_regression_models/ridge_model_alpha_1.0.pkl
Model with alpha 10.0 saved as linear_regression_models/ridge_model_alpha_10.0.pk
l
Model with alpha 100.0 saved as linear_regression_models/ridge_model_alpha_100.0.
pkl
Model with alpha 1000.0 saved as linear_regression_models/ridge_model_alpha_1000.
0.pkl
```

In [17]:
```python
# Get the best alpha value
best_alpha = grid_search.best_params_['alpha']
print(f"Best alpha found: {best_alpha}")
```

```
# Get the best model
best_model = grid_search.best_estimator_
```

```
Best alpha found: 1000
```

In [18]:

```
# Predict on the validation set
y_pred = best_model.predict(X_val_scaled)

# Calculate MSE for the validation set
mse = mean_squared_error(y_val, y_pred)
print(f"Mean Squared Error on Validation Set with Best Alpha: {mse:.2f}")
```

```
Mean Squared Error on Validation Set with Best Alpha: 1027.29
```

In [ ]:

```
```