In [9]:

```
# Importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
```

In [10]:

```
# Load the dataset
data = pd.read_csv(r'C:\Users\HP\Downloads\Final_pre_processing_data.cs
print(data)
```

```
     Unnamed: 0   id  age  bp  sg  al  su  rbc  pc  pcc  ...  pcv  wc
rc  \
0             0    0   40   3   3   1   0    1   1    0  ...   30  69
33
1             1    1    5   0   3   4   0    1   1    0  ...   24  53
33
2             2    2   54   3   1   2   3    1   1    0  ...   17  67
33
3             3    3   40   2   0   4   0    1   0    1  ...   18  59
18
4             4    4   43   3   1   2   0    1   1    0  ...   21  65
26
..          ...  ...  ...  ..  ..  ..  ..  ...  ..  ...  ...  ...  ..
..
395         395  395   47   3   3   0   0    1   1    0  ...   33  59
29
396         396  396   34   2   4   0   0    1   1    0  ...   40  69
43
397         397  397    8   3   3   0   0    1   1    0  ...   35  58
35
398         398  398   11   1   4   0   0    1   1    0  ...   37  64
40
399         399  399   50   3   4   0   0    1   1    0  ...   39  60
42

     htn  dm  cad  appet  pe  ane  classification
0      1   2    0      0   0    0               0
1      0   1    0      0   0    0               0
2      0   2    0      1   0    1               0
3      1   1    0      1   1    1               0
4      0   1    0      0   0    0               0
..   ...  ..  ...    ...  ..  ...             ...
395    0   1    0      0   0    0               1
396    0   1    0      0   0    0               1
397    0   1    0      0   0    0               1
398    0   1    0      0   0    0               1
399    0   1    0      0   0    0               1

[400 rows x 27 columns]
```

In [11]:

```
# Display the first few rows of the dataset
data.head()
```

Out[11]:

	Unnamed: 0	id	age	bp	sg	al	su	rbc	pc	pcc	...	pcv	wc	rc	htn	dm	cad	a
0	0	0	40	3	3	1	0	1	1	0	...	30	69	33	1	2	0	
1	1	1	5	0	3	4	0	1	1	0	...	24	53	33	0	1	0	
2	2	2	54	3	1	2	3	1	1	0	...	17	67	33	0	2	0	
3	3	3	40	2	0	4	0	1	0	1	...	18	59	18	1	1	0	
4	4	4	43	3	1	2	0	1	1	0	...	21	65	26	0	1	0	

5 rows × 27 columns

In [34]:

```
# Data preprocessing
# Handling missing values
data = data.replace("?", pd.NA)
data = data.dropna()
print("\nData Preprocessing:\n",data)
print("Missing values handled.")
```

```
Data Preprocessing:
      Unnamed: 0   id  age  bp  sg  al  su  rbc  pc  pcc  ...  pcv  wc  
rc  \
0              0    0   40   3   3   1   0    1   1    0  ...   30   69  
33  
1              1    1    5   0   3   4   0    1   1    0  ...   24   53  
33  
2              2    2   54   3   1   2   3    1   1    0  ...   17   67  
33  
3              3    3   40   2   0   4   0    1   0    1  ...   18   59  
18  
4              4    4   43   3   1   2   0    1   1    0  ...   21   65  
26  
..           ...  ...  ...  ..  ..  ..  ..  ...  ..  ...  ...  ...  ..  
..  
395          395  395   47   3   3   0   0    1   1    0  ...   33   59  
29  
396          396  396   34   2   4   0   0    1   1    0  ...   40   69  
43  
397          397  397    8   3   3   0   0    1   1    0  ...   35   58  
35  
398          398  398   11   1   4   0   0    1   1    0  ...   37   64  
40  
399          399  399   50   3   4   0   0    1   1    0  ...   39   60  
42  

     htn  dm  cad  appet  pe  ane  classification  
0      1   2    0      0   0    0               0  
1      0   1    0      0   0    0               0  
2      0   2    0      1   0    1               0  
3      1   1    0      1   1    1               0  
4      0   1    0      0   0    0               0  
..   ...  ..  ...    ...  ..  ...             ...  
395    0   1    0      0   0    0               1  
396    0   1    0      0   0    0               1  
397    0   1    0      0   0    0               1  
398    0   1    0      0   0    0               1  
399    0   1    0      0   0    0               1  

[400 rows x 27 columns]
Missing values handled.
```

In [35]:

```
# Encoding categorical variables
label_encoder = LabelEncoder()
for column in data.columns:
    if data[column].dtype == 'object':
        data[column] = label_encoder.fit_transform(data[column])
```

In [42]:

```
# Splitting the dataset into features and target variable
X = data.drop('classification', axis=1)
y = data['classification']
print("\nX Values:\n")
print(X)
print("\ny Values:\n")
print(y)
```

```
X Values:

     Unnamed: 0   id  age  bp  sg  al  su  rbc  pc  pcc  ...  hemo  pc
v  wc  \
0             0    0   40   3   3   1   0    1   1    0  ...    90   3
0  69
1             1    1    5   0   3   4   0    1   1    0  ...    49   2
4  53
2             2    2   54   3   1   2   3    1   1    0  ...    32   1
7  67
3             3    3   40   2   0   4   0    1   0    1  ...    48   1
8  59
4             4    4   43   3   1   2   0    1   1    0  ...    52   2
1  65
..          ...  ...  ...  ..  ..  ..  ..  ...  ..  ...  ...   ...   
...  ..
395         395  395   47   3   3   0   0    1   1    0  ...    93   3
3  59
396         396  396   34   2   4   0   0    1   1    0  ...   101   4
0  69
397         397  397    8   3   3   0   0    1   1    0  ...    94   3
5  58
398         398  398   11   1   4   0   0    1   1    0  ...    78   3
7  64
399         399  399   50   3   4   0   0    1   1    0  ...    94   3
9  60

     rc  htn  dm  cad  appet  pe  ane
0    33    1   2    0      0   0    0
1    33    0   1    0      0   0    0
2    33    0   2    0      1   0    1
3    18    1   1    0      1   1    1
4    26    0   1    0      0   0    0
..   ..  ...  ..  ...    ...  ..  ...
395  29    0   1    0      0   0    0
396  43    0   1    0      0   0    0
397  35    0   1    0      0   0    0
398  40    0   1    0      0   0    0
399  42    0   1    0      0   0    0

[400 rows x 26 columns]

y Values:

0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    1
399    1
Name: classification, Length: 400, dtype: int64
```

In [47]:

```
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
print("\nX training data:\n")
print(X_train)
print("\nX testing data:\n")
print(X_test)
print("\ny training data:\n")
print(y_train)
print("\ny testing data:\n")
print(y_test)
```

```
X training data:

     Unnamed: 0   id  age  bp  sg  al  su  rbc  pc  pcc  ...  hemo  pc
v  wc  \
3             3    3   40   2   0   4   0    1   0    1  ...    48   1
8  59
18           18   18   52   5   4   0   3    1   1    0  ...    63   2
3  10
202         202  202   69   1   3   0   0    1   1    0  ...    17   1
0  87
250         250  250   32   3   4   0   0    1   1    0  ...    86   3
4   2
274         274  274   12   3   3   0   0    1   1    0  ...    80   3
0  87
..          ...  ...  ...  ..  ..  ..  ..  ...  ..  ...  ...   ...
...  ..
71           71   71   38   1   1   1   0    1   1    0  ...    34   1
4  24
106         106  106   42   4   3   0   0    1   1    0  ...     5
3  57
270         270  270   16   3   4   0   0    1   1    0  ...    79   2
7  64
348         348  348   30   3   3   0   0    1   1    0  ...    72   3
0  65
102         102  102   11   1   1   0   0    1   1    0  ...    75   3
8  62

     rc  htn  dm  cad  appet  pe  ane
3    18    1   1    0      1   1    1
18   23    1   2    1      0   0    0
202  33    0   2    0      0   0    1
250  25    0   1    0      0   0    0
274  33    0   1    0      0   0    0
..   ..  ...  ..  ...    ...  ..  ...
71   11    1   2    0      0   0    0
106  33    1   2    0      0   1    1
270  31    0   1    0      0   0    0
348  45    0   1    0      0   0    0
102  33    0   1    0      0   0    0

[320 rows x 26 columns]

X testing data:

     Unnamed: 0   id  age  bp  sg  al  su  rbc  pc  pcc  ...  hemo  pc
v  wc  \
209         209  209   12   2   3   0   0    1   1    0  ...    51   2
7  61
280         280  280   39   3   3   0   0    1   1    0  ...    69   3
8  72
33           33   33   52   5   3   2   0    0   0    0  ...    37   1
5  87
210         210  210   51   5   2   4   2    1   1    0  ...    12
6  87
93           93   93   65   5   1   3   2    0   0    1  ...    28   1
6  62
..          ...  ...  ...  ..  ..  ..  ..  ...  ..  ...  ...   ...
...  ..
246         246  246   40   6   2   3   0    0   1    1  ...    23   1
2  43
```

```
227         227  227   49   3   2   0   0    1   1    0  ...    49   2
2  64
369         369  369   67   2   3   0   0    1   1    0  ...    72   3
2   1
176         176  176   14   4   1   4   0    1   0    1  ...    20
9  18
289         289  289   34   2   3   0   0    1   1    0  ...   102   2
9  63

     rc  htn  dm  cad  appet  pe  ane
209  33    0   1    0      0   0    0
280  33    0   1    0      0   0    0
33   33    1   1    0      1   0    0
210  18    1   2    1      0   0    1
93   11    1   2    1      1   0    0
..   ..  ...  ..  ...    ...  ..  ...
246   3    1   1    1      0   0    1
227  17    1   2    0      0   0    0
369  28    0   1    0      0   0    0
176  18    0   1    0      0   0    1
289  34    0   1    0      0   0    0

[80 rows x 26 columns]

y training data:

3      0
18     0
202    0
250    1
274    1
      ..
71     0
106    0
270    1
348    1
102    0
Name: classification, Length: 320, dtype: int64

y testing data:

209    0
280    1
33     0
210    0
93     0
      ..
246    0
227    0
369    1
176    0
289    1
Name: classification, Length: 80, dtype: int64
```

In [48]:

```
# Training the models
models = {
    'Random Forest': RandomForestClassifier(),
    'Logistic Regression': LogisticRegression()
}
```

In [52]:

```
# Dictionary to store accuracy scores
accuracy_scores = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(y_pred)
    accuracy = accuracy_score(y_test, y_pred)*100
    print(accuracy)
    accuracy_scores[name] = accuracy
```

```
[0 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0
0 0
 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0
0 0
 1 0 0 1 0 1]
100.0
[0 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0
0 0
 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0
0 0
 1 0 0 1 0 1]
97.5

C:\Users\HP\anaconda3\Lib\site-packages\sklearn\linear_model\_logisti
c.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as show
n in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic
-regression (https://scikit-learn.org/stable/modules/linear_model.html
#logistic-regression)
  n_iter_i = _check_optimize_result(
```

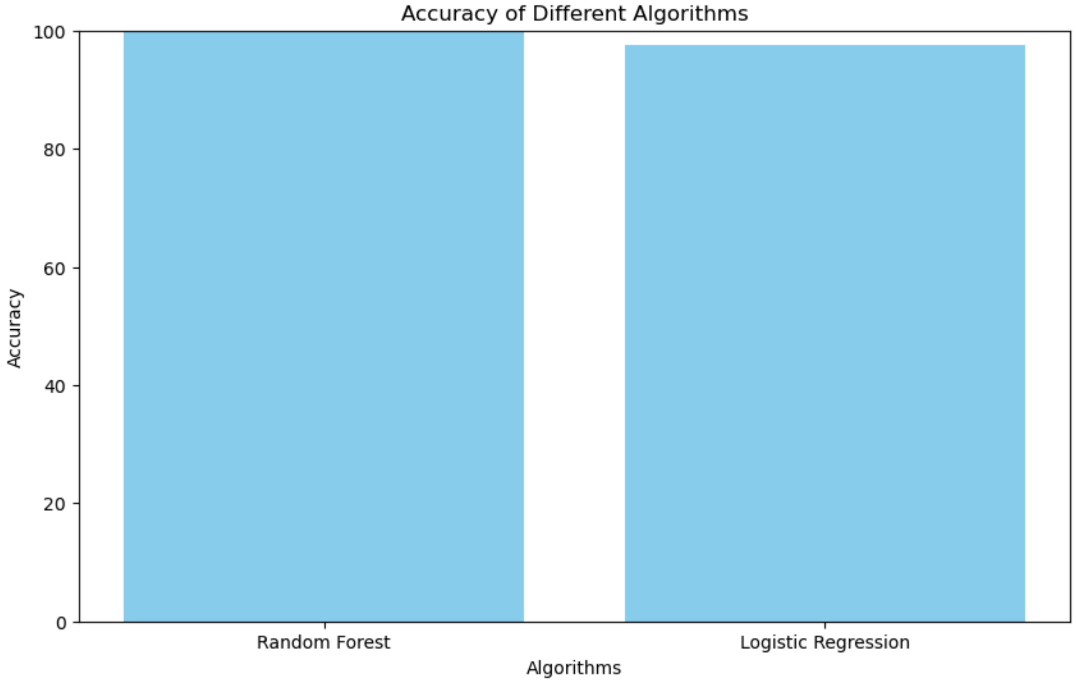
In [29]:

```
# Comparing accuracy scores
print("Accuracy Scores:")
for name, accuracy in accuracy_scores.items():
    print(f"{name}: {accuracy:.2f}%")
```

```
Accuracy Scores:
Random Forest: 100.00%
Logistic Regression: 97.50%
```

In [30]:

```
# Plotting the accuracy scores
plt.figure(figsize=(10, 6))
plt.bar(accuracy_scores.keys(), accuracy_scores.values(), color='skyblue
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Algorithms')
plt.ylim(0, 100)  # Set y-axis limit to percentage range
plt.show()
```



In [53]:

```
# Predicting if the patient has the disease or not
# We'll use the model with the highest accuracy
best_model_name = max(accuracy_scores, key=accuracy_scores.get)
best_model = models[best_model_name]
print(best_model)
```

```
RandomForestClassifier()
```

In [56]:

```
# Predicting for a new patient
new_patient_data = [[11,11,55,2,1,3,0,0,0,1,0,137,46,23,15,15,44,18,40,
]]
prediction = best_model.predict(new_patient_data)
prediction_probability = best_model.predict_proba(new_patient_data)
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarni
ng: X does not have valid feature names, but RandomForestClassifier wa
s fitted with feature names
  warnings.warn(
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarni
ng: X does not have valid feature names, but RandomForestClassifier wa
s fitted with feature names
  warnings.warn(
```

In [24]:

```
print("\nPrediction:")
if prediction[0] == 1:
    print("The patient has chronic kidney disease.")
else:
    print("The patient does not have chronic kidney disease.")

print("\nPrediction Probability:")
print(f"Probability of not having chronic kidney disease: {prediction_pr
print(f"Probability of having chronic kidney disease: {prediction_probab
```

```

Prediction:
The patient does not have chronic kidney disease.

Prediction Probability:
Probability of not having chronic kidney disease: 100.00%
Probability of having chronic kidney disease: 0.00%
```

In [25]:

```
# Predicting for a new patient
new_patient_data = [[334,334,17,3,4,0,0,1,1,0,0,52,32,8,20,8,90,29,49,25

]]
prediction = best_model.predict(new_patient_data)
prediction_probability = best_model.predict_proba(new_patient_data)
```

```
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarni
ng: X does not have valid feature names, but RandomForestClassifier wa
s fitted with feature names
  warnings.warn(
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarni
ng: X does not have valid feature names, but RandomForestClassifier wa
s fitted with feature names
  warnings.warn(
```

In [26]:

```
print("\nPrediction:")
if prediction[0] == 1:
    print("The patient has chronic kidney disease.")
else:
    print("The patient does not have chronic kidney disease.")

print("\nPrediction Probability:")
print(f"Probability of not having chronic kidney disease: {prediction_pr
print(f"Probability of having chronic kidney disease: {prediction_probab
```

```

Prediction:
The patient has chronic kidney disease.

Prediction Probability:
Probability of not having chronic kidney disease: 0.00%
Probability of having chronic kidney disease: 100.00%
```

In [ ]:

```
```