Implement K-Nearest Neighbors algorithm on social network dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset. Link to Dataset: https://www.kaggle.com/datasets/rakeshrau/social-network-ads The dataset contains the details of users in a social networking site to find whether a user buys a product by clicking the ad on the site based on their salary, age, and gender.

In [19]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
import seaborn as sns
import matplotlib.pyplot as plt
```

In [20]:
```
# Load the dataset
df = pd.read_csv("C:/Users/samik/Downloads/Social_Network_Ads.csv")
print(df.head(10))  # View the first few rows
```

```
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
5  15728773    Male   27            58000          0
6  15598044  Female   27            84000          0
7  15694829  Female   32           150000          1
8  15600575    Male   25            33000          0
9  15727311  Female   35            65000          0
```

In [21]:
```
# Preprocessing: Encode 'Gender' and separate features and target
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])  # Female=0, Male=1
```

In [22]:
```
# Select features and target variable
X = df[['Age', 'EstimatedSalary', 'Gender']]
y = df['Purchased']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [23]:
```
# Initialize and train the KNN model
knn = KNeighborsClassifier(n_neighbors=5)  # You may tune 'n_neighbors' for optimal results
knn.fit(X_train, y_train)
```

Out[23]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

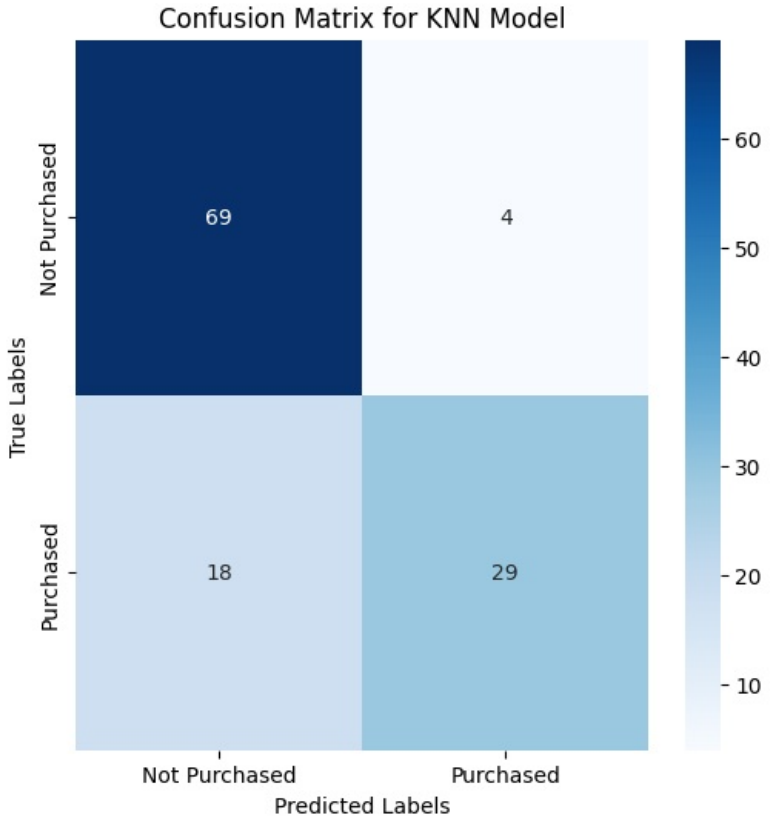
In [24]:
```
# Make predictions on the test set
y_pred = knn.predict(X_test)
```

In [25]:
```
# Calculate performance metrics
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

In [26]:
```
# Display results
print("Confusion Matrix:\n", conf_matrix)
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Confusion Matrix:
 [[69  4]
 [18 29]]
Accuracy: 0.82
Error Rate: 0.18
Precision: 0.88
Recall: 0.62
```

In [27]:
```
# Plot confusion matrix as heatmap
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=['Not Purchased', 'Purchased'], yticklal
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix for KNN Model")
plt.show()
```



In [ ]: