Load data and initial exploration

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from datetime import datetime

# Load the data
df = pd.read_csv('/content/intrenship_data.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5811 entries, 0 to 5810
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   application_id          5811 non-null   int64  
 1   first_name              5811 non-null   object 
 2   last_name               5811 non-null   object 
 3   email                   5811 non-null   object 
 4   phone                   5811 non-null   object 
 5   university              5811 non-null   int64  
 6   major                   5811 non-null   int64  
 7   gpa                     5811 non-null   float64
 8   graduation_year         5811 non-null   int64  
 9   submission_date         5811 non-null   object 
 10  skills                  5811 non-null   object 
 11  projects                4432 non-null   object 
 12  ip_address              5811 non-null   object 
 13  submission_device       5811 non-null   int64  
 14  time_spent_on_form      5811 non-null   int64  
 15  is_anomaly              5811 non-null   int64  
 16  anomaly_type            430 non-null    object 
 17  submission_hour         5811 non-null   int64  
 18  submission_day_of_week  5811 non-null   int64  
dtypes: float64(1), int64(9), object(9)
memory usage: 862.7+ KB
```

In [4]:

```
df.head()
```

Out[4]:

	application_id	first_name	last_name	email	phone	university	major	gpa	graduation_y
0	1411	Wyatt	Baker	wyatt.baker79@gmail.com	284.525.2357	61	6	3.45	2
1	3462	Erika	Ruiz	erika.ruiz43@hotmail.com	602.294.0795	37	5	3.96	2
2	1101	Logan	Davidson	logan.davidson19@hotmail.com	001-264-740-1137x1308	20	8	2.83	2
3	1333	Alicia	Harris	alicia.harris94@yahoo.com	270.319.2564x69677	37	1	3.46	2
4	1638	Michael	Bates	michael.bates56@gmail.com	350.722.6922x7910	20	8	3.54	2

In [5]:

```
print(df['is_anomaly'].value_counts())
```

```
is_anomaly
0    5381
1     430
Name: count, dtype: int64
```

In [6]:
```
print(df['anomaly_type'].value_counts())
```

```
anomaly_type
duplicate            86
bot_pattern          77
rapid_submission     74
inconsistent_data    70
fake_university      63
impossible_gpa       60
Name: count, dtype: int64
```

Data preprocessing

In [9]:
```
# Convert submission_date to datetime
df['submission_date'] = pd.to_datetime(df['submission_date'], errors='coerce')
df.dropna(subset=['submission_date'], inplace=True)

# Now proceed with feature extraction
df['submission_hour'] = df['submission_date'].dt.hour
df['submission_minute'] = df['submission_date'].dt.minute
df['submission_day'] = df['submission_date'].dt.day

# Create a feature for submission speed (time spent on form)
df['submission_speed'] = df['time_spent_on_form'] / df['skills'].str.split(',').str.len()

# Create binary features from skills
skills_list = ['Java', 'Python', 'SQL', 'C++', 'Web Development',
               'Machine Learning', 'Data Analysis', 'Cloud Computing',
               'Project Management', 'Communication']
for skill in skills_list:
    df[f'skill_{skill}'] = df['skills'].str.contains(skill).astype(int)

# Handle missing values
df.fillna({'projects': '', 'anomaly_type': 'normal'}, inplace=True)
```

Rule base anamoly detection

In [10]:
```
# Detect duplicates based on email, phone, and IP
df['duplicate_email'] = df.duplicated('email', keep=False)
df['duplicate_phone'] = df.duplicated('phone', keep=False)
df['duplicate_ip'] = df.duplicated('ip_address', keep=False)

# Detect rapid submissions (less than 10 seconds)
df['rapid_submission'] = df['time_spent_on_form'] < 10

# Detect impossible GPA values
df['impossible_gpa'] = (df['gpa'] < 0) | (df['gpa'] > 4)

# Detect inconsistent graduation year (before submission year)
df['inconsistent_graduation'] = df['graduation_year'] < df['submission_date'].dt.year

# Detect fake universities (assuming university IDs should be within a certain range)
df['fake_university'] = ~df['university'].between(1, 100)

# Combine rule-based anomalies
df['rule_based_anomaly'] = (df['duplicate_email'] | df['duplicate_phone'] |
                            df['duplicate_ip'] | df['rapid_submission'] |
                            df['impossible_gpa'] | df['inconsistent_graduation'] |
                            df['fake_university'])
```

Machine Learning Anomaly Detection

Isolation Forest

In [11]:
```
# Prepare features for Isolation Forest
features = ['gpa', 'time_spent_on_form', 'submission_hour', 'submission_day_of_week'] + \
           [f'skill_{skill}' for skill in skills_list]

X = df[features]

# Train Isolation Forest
```

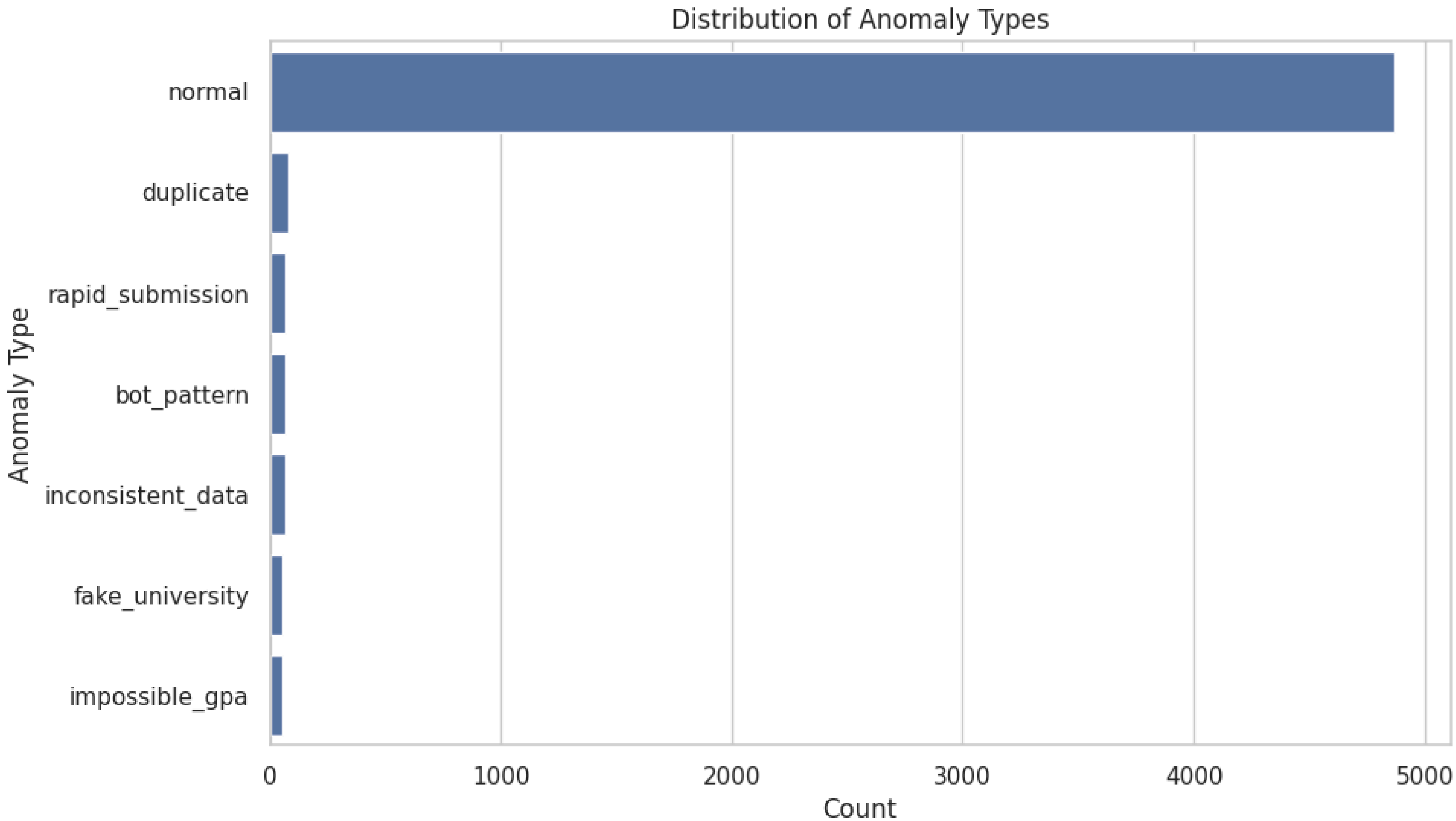
```
iso_forest = IsolationForest(contamination=0.05, random_state=42)
iso_forest.fit(X)
df['iso_anomaly'] = iso_forest.predict(X)
df['iso_anomaly_score'] = iso_forest.decision_function(X)

# Convert to binary (1 = normal, -1 = anomaly)
df['iso_anomaly'] = df['iso_anomaly'].apply(lambda x: 1 if x == -1 else 0)
```

K-Means Clustering

In [12]:
```
# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means
kmeans = KMeans(n_clusters=5, random_state=42)
df['cluster'] = kmeans.fit_predict(X_scaled)

# Calculate distance to cluster centers
distances = kmeans.transform(X_scaled)
df['cluster_distance'] = distances.min(axis=1)

# Flag anomalies as points far from their cluster center
df['kmeans_anomaly'] = (df['cluster_distance'] > df['cluster_distance'].quantile(0.95)).astype(int)
```

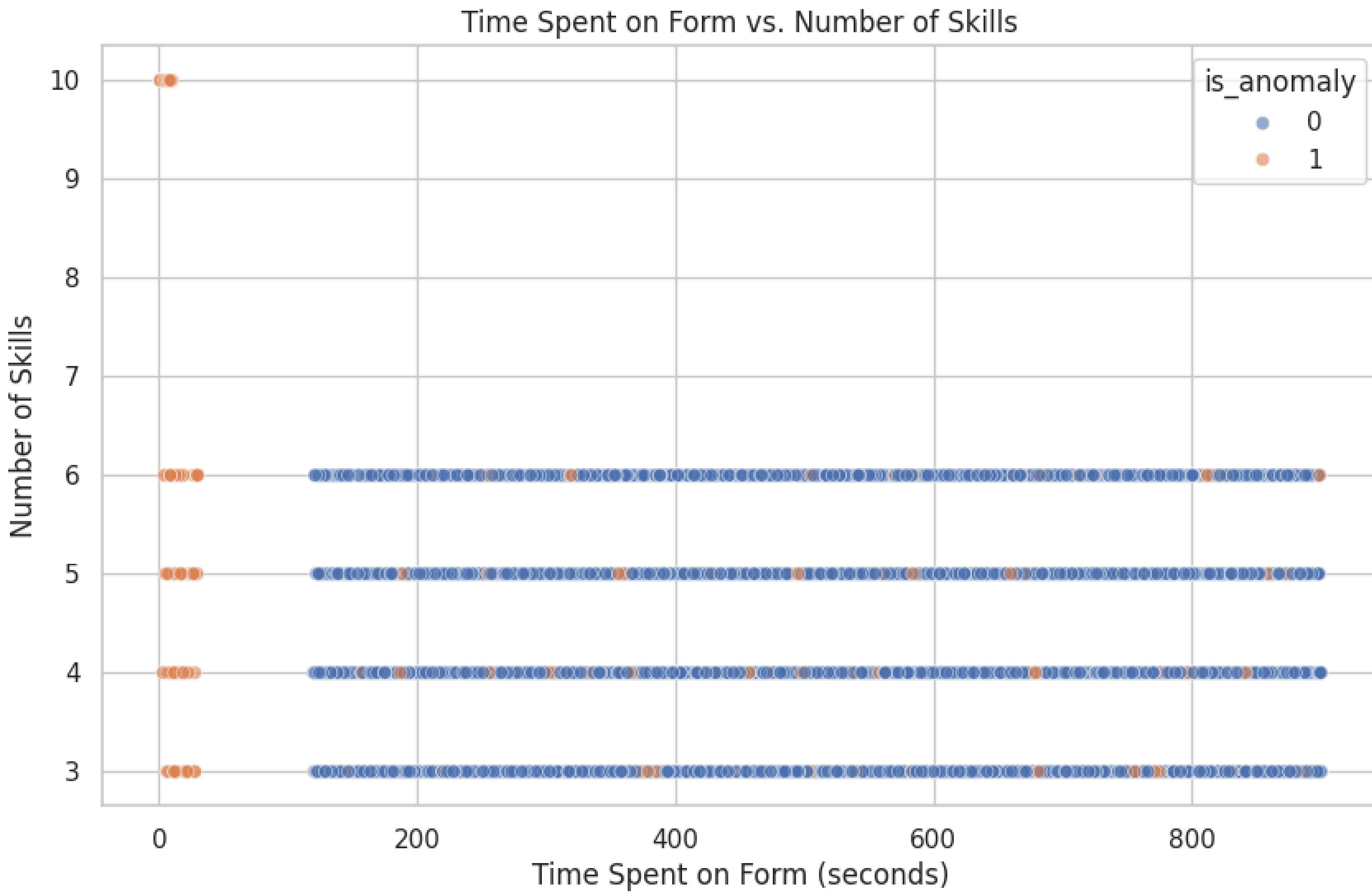
Visualization

In [18]:
```
# Set style
sns.set(style="whitegrid")

# Plot anomaly types
plt.figure(figsize=(10, 6))
sns.countplot(y='anomaly_type', data=df, order=df['anomaly_type'].value_counts().index)
plt.title('Distribution of Anomaly Types')
plt.xlabel('Count')
plt.ylabel('Anomaly Type')
plt.show()
```

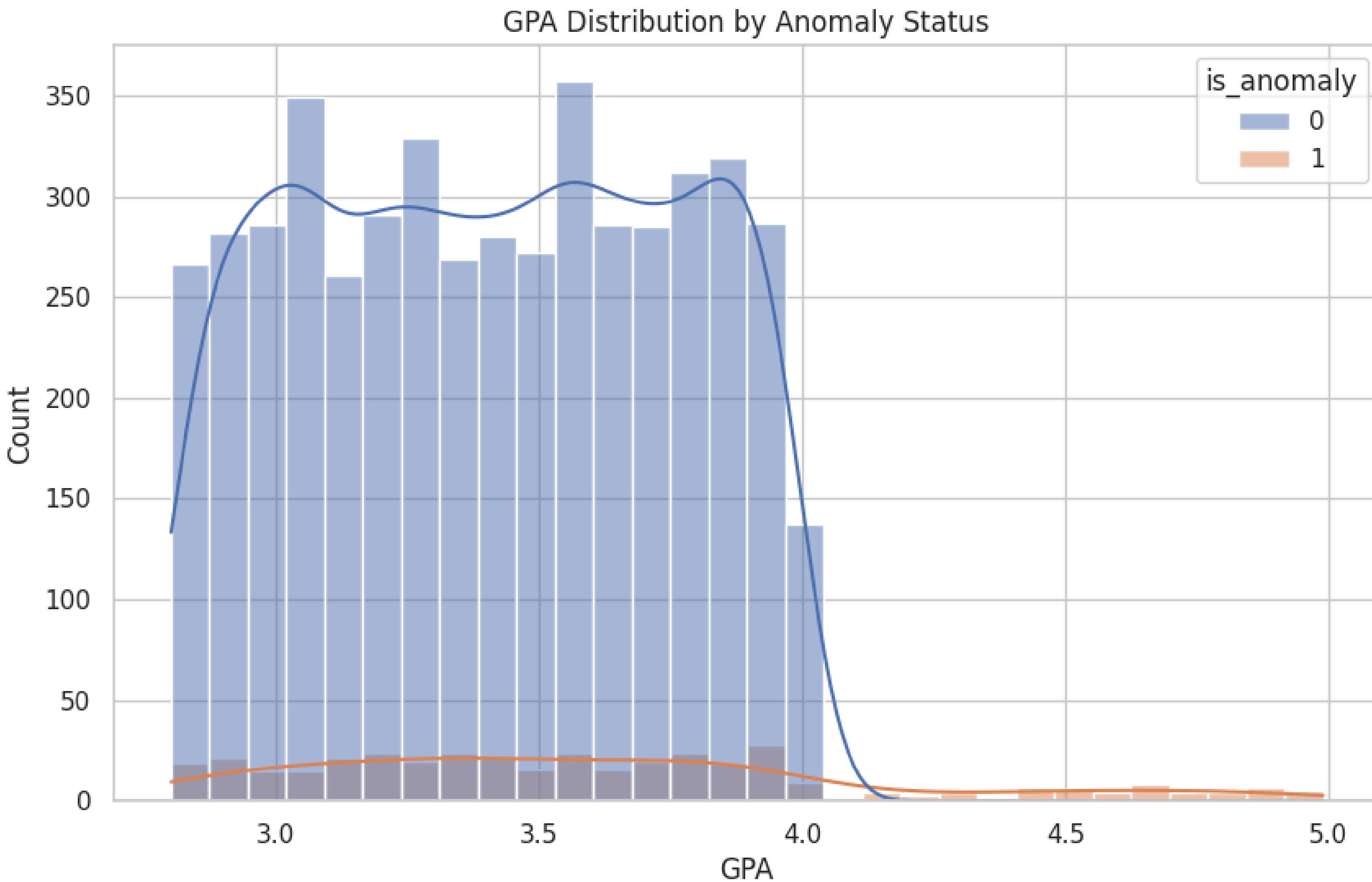


In [14]:

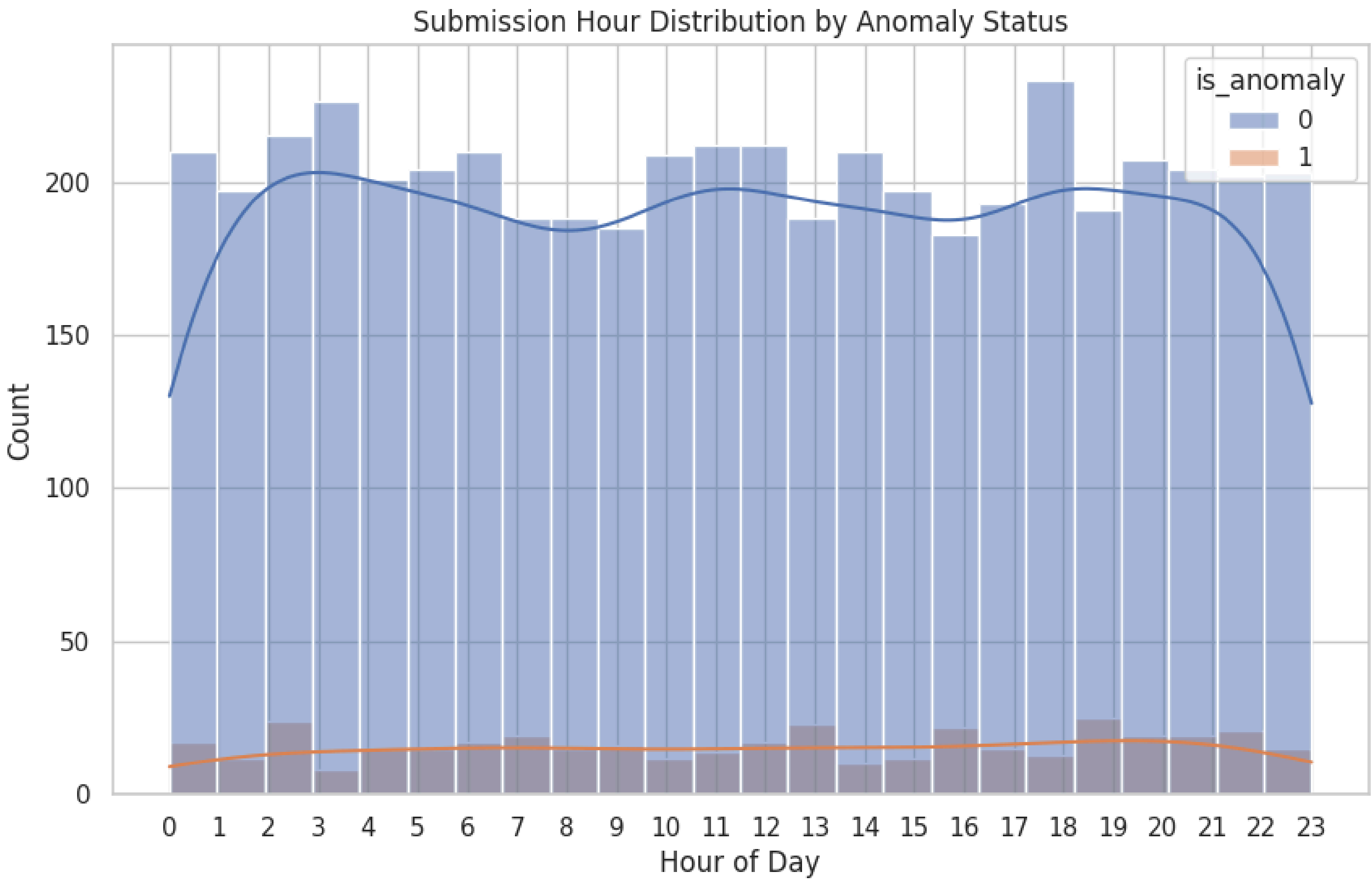
```
# Plot time spent on form vs. number of skills
plt.figure(figsize=(10, 6))
sns.scatterplot(x='time_spent_on_form', y=df['skills'].str.split(',').str.len(),
                hue='is_anomaly', data=df, alpha=0.6)
plt.title('Time Spent on Form vs. Number of Skills')
plt.xlabel('Time Spent on Form (seconds)')
plt.ylabel('Number of Skills')
plt.show()
```



In [15]:
```
# Plot GPA distribution
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='gpa', hue='is_anomaly', bins=30, kde=True)
plt.title('GPA Distribution by Anomaly Status')
plt.xlabel('GPA')
plt.ylabel('Count')
plt.show()
```

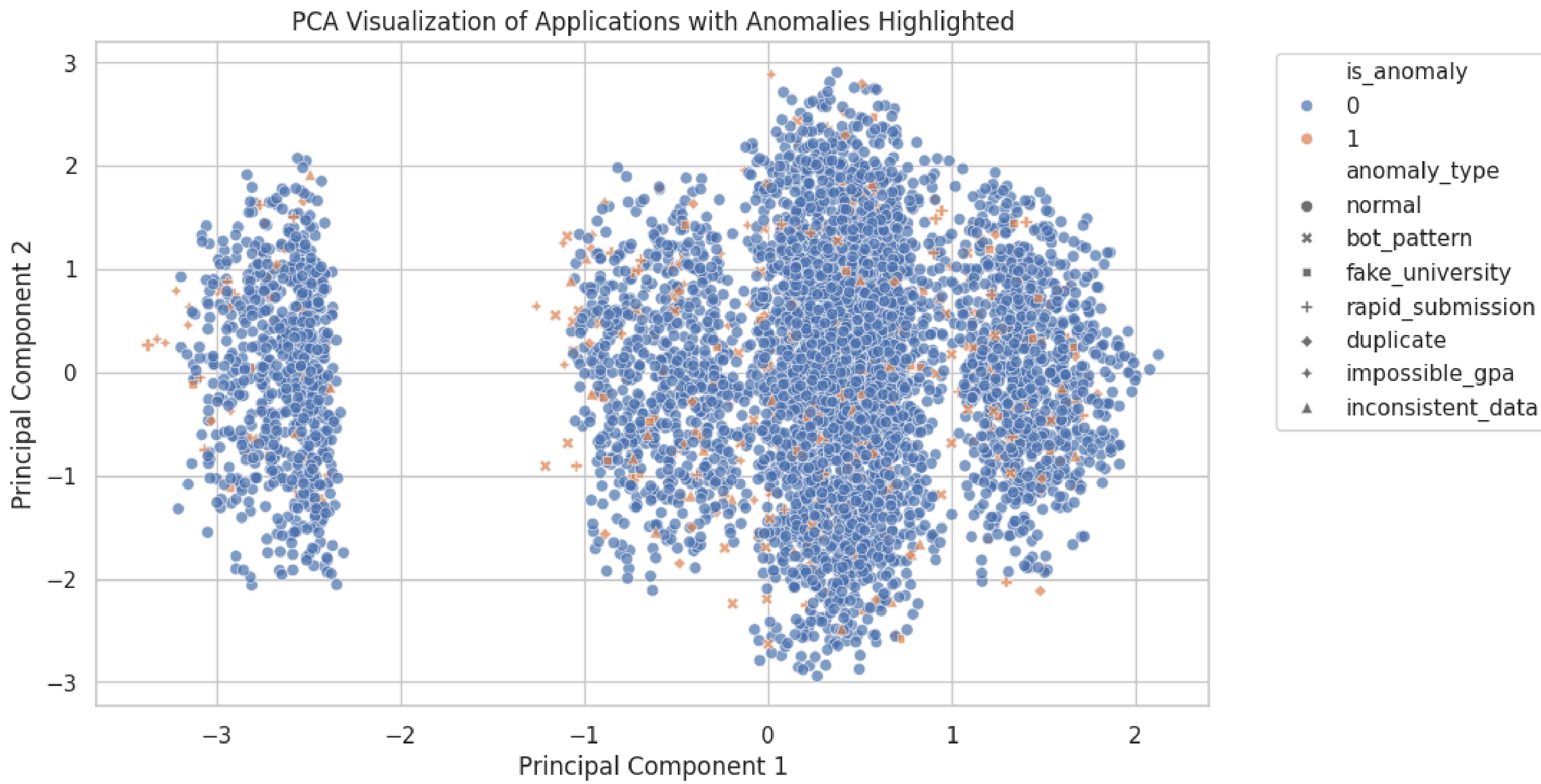


In [16]:
```
# Plot submission hour distribution
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='submission_hour', hue='is_anomaly', bins=24, kde=True)
plt.title('Submission Hour Distribution by Anomaly Status')
plt.xlabel('Hour of Day')
plt.ylabel('Count')
plt.xticks(range(24))
plt.show()
```



In [17]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df['is_anomaly'],
                style=df['anomaly_type'], alpha=0.7)
```

```
plt.title('PCA Visualization of Applications with Anomalies Highlighted')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



6. Alert System Design

In [19]:
```
def generate_alerts(row):
    alerts = []

    # Rule-based alerts
    if row['duplicate_email']:
        alerts.append("Duplicate email detected")
```

```python
    if row['duplicate_phone']:
        alerts.append("Duplicate phone number detected")
    if row['duplicate_ip']:
        alerts.append("Duplicate IP address detected")
    if row['rapid_submission']:
        alerts.append(f"Extremely rapid submission ({row['time_spent_on_form']} seconds)")
    if row['impossible_gpa']:
        alerts.append(f"Impossible GPA value ({row['gpa']})")
    if row['inconsistent_graduation']:
        alerts.append(f"Inconsistent graduation year ({row['graduation_year']})")
    if row['fake_university']:
        alerts.append(f"Suspicious university ID ({row['university']})")

    # ML-based alerts
    if row['iso_anomaly'] == 1:
        alerts.append("Isolation Forest anomaly detected")
    if row['kmeans_anomaly'] == 1:
        alerts.append("K-Means anomaly detected (far from cluster center)")

    return alerts if alerts else ["No alerts"]

# Apply alert generation
df['alerts'] = df.apply(generate_alerts, axis=1)

# Display suspicious applications
suspicious = df[df['alerts'].apply(lambda x: x != ['No alerts'])]
print(f"Found {len(suspicious)} suspicious applications out of {len(df)} total")

# Save suspicious applications to CSV
suspicious.to_csv('suspicious_applications.csv', index=False)
```

```
Found 2846 suspicious applications out of 5263 total
```

📊 Key Insights from Visualizations

🔎 Anomaly Type Distribution

- **Most Common Anomalies:**
  - `bot_pattern` (repetitive entries)

    - `rapid_submission`
    - `fake_university`
- **Less Common but Significant:**
    - `impossible_gpa`
    - `inconsistent_data`

⏱️ Time Spent vs. Skills

- Normal applications:
    - Positive correlation between **time spent** and **number of skills**.
- Anomalous applications:
    - Extremely short times for many skills.
    - Unusually long times for very few skills.

🎓 GPA Distribution

- Most GPAs are clustered between **2.5 and 4.0**.
- Anomalies:
    - Values outside the **0-4** range.
    - Some extreme values within the valid range.

🕒 Submission Times

- Normal applications:
    - Follow a typical daily pattern (peaks during working hours).
- Anomalous applications:
    - Evenly spread throughout the day.
    - Some concentration at **unusual hours**.

🧩 Cluster Visualization

- Normal applications:
  - Form **tight clusters** on the PCA plot.
- Anomalous applications:
  - Appear as **outliers** scattered away from main clusters.

✅ Recommendations

🚨 Automated Alert System

- Implement **real-time monitoring** to flag applications with multiple anomaly indicators.
- **Prioritize alerts** based on severity:
  - `impossible values > duplicates > ML anomalies`

👁️ Review Process

- Create a **dashboard** for human reviewers:
  - Display flagged applications with full details.
  - Show anomaly scores for transparency.

♻️ Continuous Improvement

- **Update ML models** regularly with new data and confirmed fraud cases.
- **Adjust rule thresholds** based on historical patterns and trends.

🛡️ Preventive Measures

- Implement **CAPTCHA** or other bot-prevention mechanisms.

- **Validate university IDs** against a known list.
- **Rate-limit submissions** from the same IP address.

> This **hybrid approach** (combining rule-based detection and ML) ensures **effective identification** of suspicious internship applications while minimizing **false positives**.

In [ ]:

```
```