```
!pip install pandas vaderSentiment gensim
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 3.6 MB/s eta 0:00:00
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.10.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim) (6.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.2
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2023.7.22
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

```
import requests
from bs4 import BeautifulSoup
import csv
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import gensim
from gensim import corpora
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
import re
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import seaborn as sns


# Initialize NLTK stopwords
nltk.download('stopwords')
nltk.download('punkt')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
#scraping quran data and storing in csv file according to pillars
base_url = "https://www.clearquran.com/"
pillars = {
    "Shahada": ["faith", "testimony", "witness", "belief"],
    "Salat": ["prayer", "friday", "worship", "ritual"],
    "Zakat": ["charity", "almsgiving", "poor"],
    "Sawm": ["fasting", "Ramadan", "abstain"],
    "Hajj": ["pilgrimage", "Mecca", "Kaaba", "Hajj", "Umrah"]
}

# Create a CSV file to store the results
with open('quran_ayahs.csv', 'w', newline='', encoding='utf-8') as csvfile:
    fieldnames = ['Surah Number', 'Surah Name', 'Ayah Number', 'Ayah Text', 'Pillar']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    # Iterate over surahs from 1 to 114
    for surah_number in range(1, 115):
        # Construct the URL for the surah page
        surah_url = f"{base_url}{surah_number:03d}.html"

        # Fetch the website content
        response = requests.get(surah_url)
        if response.status_code == 200:
            soup = BeautifulSoup(response.content, 'html.parser')

            # Extract the surah name from the page title
            title_text = soup.title.text.strip()
            surah_name = title_text.split('.')[2].strip()

            # quran text
            text = soup.find('div', {'id': 'quran-text'})

            # getting ayahs
            ayahs = text.find_all('p')

            # Store surah, ayah data, and pillar in the CSV file
            for ayah_number, ayah in enumerate(ayahs, start=1):
                ayah_text = ayah.get_text(strip=True)
                ayah_text_lower = ayah_text.lower()
                pillar = None

                # Determine the pillar for the ayah based on keywords
                for pillar_name, words in pillars.items():
                    if any(word in ayah_text_lower for word in words):
                        pillar = pillar_name
                        break

                writer.writerow({
                    'Surah Number': surah_number,
                    'Surah Name': surah_name,
                    'Ayah Number': ayah_number,
                    'Ayah Text': ayah_text,
                    'Pillar': pillar
                })
print("Data has been stored in the CSV file.")
```

```
Data has been stored in the CSV file.
```

```
#scraping sunnah data and storing in csv file according to pillars
def scrape_hadiths(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.content, "html.parser")

    hadiths = []

    for hadith in soup.find_all("div", class_="actualHadithContainer"):
        hadith_text = hadith.find("div", class_="text_details").get_text().strip()
        hadiths.append(hadith_text)

    return hadiths

pillar_keywords = {
    "Shahada": ["faith", "testimony", "witness", "belief"],
    "Salat": ["prayer", "friday", "worship", "ritual"],
    "Zakat": ["charity", "almsgiving", "poor"],
    "Sawm": ["fasting", "Ramadan", "abstain"],
    "Hajj": ["pilgrimage", "Mecca", "Kaaba", "Hajj", "Umrah"]
}

hadith_collections = [
    {"collection_name": "Abu Dawood", "url": "https://sunnah.com/abudawud/2"},
    {"collection_name": "Sahih Muslim", "url": "https://sunnah.com/muslim/12"},
    {"collection_name": "Sahih al-Bukhari", "url": "https://sunnah.com/bukhari/25"},
    {"collection_name": "Sahih al-Bukhari", "url": "https://sunnah.com/bukhari/30"},
    {"collection_name": "Sunan at-Tirmidhi", "url": "https://sunnah.com/tirmidhi/7"}
]

# Create a CSV file to store the results
with open('hadiths.csv', 'w', newline='', encoding='utf-8') as csvfile:
    fieldnames = ['Collection Name', 'Pillar', 'Hadith Text']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    for collection in hadith_collections:
        collection_name = collection["collection_name"]
        url = collection["url"]
        hadiths = scrape_hadiths(url)
        for hadith in hadiths:
            hadith_text_lower = hadith.lower()
            pillar = None
            for pillar_name, keywords in pillar_keywords.items():
                if any(keyword in hadith_text_lower for keyword in keywords):
                    pillar = pillar_name
                    break

            writer.writerow({
                'Collection Name': collection_name,
                'Pillar': pillar,
                'Hadith Text': hadith
            })

        print(f"Hadiths from {collection_name} categorized and stored in CSV.\n")

print("Data has been stored in the CSV file.")
```

```
Hadiths from Abu Dawood categorized and stored in CSV.

Hadiths from Sahih Muslim categorized and stored in CSV.

Hadiths from Sahih al-Bukhari categorized and stored in CSV.

Hadiths from Sahih al-Bukhari categorized and stored in CSV.

Hadiths from Sunan at-Tirmidhi categorized and stored in CSV.

Data has been stored in the CSV file.
```

```
#performing sentiment analysis and applying lda on hadith data
hadith_data = pd.read_csv("/content/hadiths.csv")
```

```
hadith_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1426 entries, 0 to 1425
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Collection Name  1426 non-null   object
 1   Pillar           669 non-null    object
 2   Hadith Text      1426 non-null   object
dtypes: object(3)
memory usage: 33.5+ KB
```

```
hadith_data.isna().sum()
```

```
Collection Name      0
Pillar             757
Hadith Text          0
dtype: int64
```

```
hadith_data = hadith_data.dropna()
```

```
# Initialize sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Preprocess the documents
def preprocess(text):
    tokens = word_tokenize(text)
    tokens = [word.lower() for word in tokens if word.isalpha() and word.lower() not in stop_words]
    return tokens

# Perform sentiment analysis and add a Sentiment column to the DataFrame
def analyze_sentiment(text):
    sentiment_score = analyzer.polarity_scores(text)
    sentiment = sentiment_score['compound']

    if sentiment >= 0.05:
        return "Positive"
    elif sentiment <= -0.05:
        return "Negative"
    else:
        return "Neutral"

hadith_data['Sentiment'] = hadith_data['Hadith Text'].apply(analyze_sentiment)

# Preprocess the documents
hadith_data['Preprocessed Text'] = hadith_data['Hadith Text'].apply(preprocess)

# Create a dictionary from the preprocessed documents
```

```
dictionary = corpora.Dictionary(hadith_data['Preprocessed Text'])

# Create a document-term matrix
doc_term_matrix = [dictionary.doc2bow(doc) for doc in hadith_data['Preprocessed Text']]

# Apply LDA model
num_topics = 5
lda_model = gensim.models.LdaModel(doc_term_matrix, num_topics=num_topics, id2word=dictionary, passes=15)

# Print the topics and their keywords
pillar_topic_mapping = {
    0: "Shahada",
    1: "Salat",
    2: "Zakat",
    3: "Sawm",
    4: "Hajj"
}

# Print the topics and their keywords
for idx, topic in lda_model.print_topics(-1):
    pillar = pillar_topic_mapping.get(idx, "Unknown")
    keywords = ", ".join(word for word in topic.split('"') if word.isalpha())
    print(f"Pillar: {pillar}")
    print(f"Keywords: {keywords}\n")
```

```
Pillar: Shahada
Keywords: prayer, allah, ﷺ, said, raised, stood, two, hands, messenger, takbir

Pillar: Salat
Keywords: allah, said, ﷺ, prayer, messenger, prophet, ibn, abu, people, came

Pillar: Zakat
Keywords: allah, would, prayer, messenger, said, ﷺ, used, one, recite, say

Pillar: Sawm
Keywords: fasting, fast, said, ﷺ, day, prophet, allah, one, month, abu

Pillar: Hajj
Keywords: prayer, said, ﷺ, prophet, allah, people, one, two, friday, charity
```

```
#performing sentiment analysis and applying lda on quran data
quran_data = pd.read_csv("/content/quran_ayahs.csv")
```

```
quran_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6348 entries, 0 to 6347
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Surah Number  6348 non-null   int64 
 1   Surah Name    6348 non-null   object
 2   Ayah Number   6348 non-null   int64 
 3   Ayah Text     6348 non-null   object
 4   Pillar        460 non-null    object
dtypes: int64(2), object(3)
memory usage: 248.1+ KB
```

```
quran_data.isna().sum()
```

```
Surah Number       0
Surah Name         0
Ayah Number        0
Ayah Text          0
Pillar          5888
dtype: int64
```

```
quran_data = quran_data.dropna()
```

```
quran_data.head(4)
```

	Surah Number	Surah Name	Ayah Number	Ayah Text	Pillar
4	1	Text, Audio, Search,	5	5.It is You we worship, and upon You we	Salat

```
# Remove numbers using regular expression
quran_data["Ayah Text"] = quran_data["Ayah Text"].apply(lambda ayah: re.sub(r'^\d+\.', '', ayah))
```

perform...

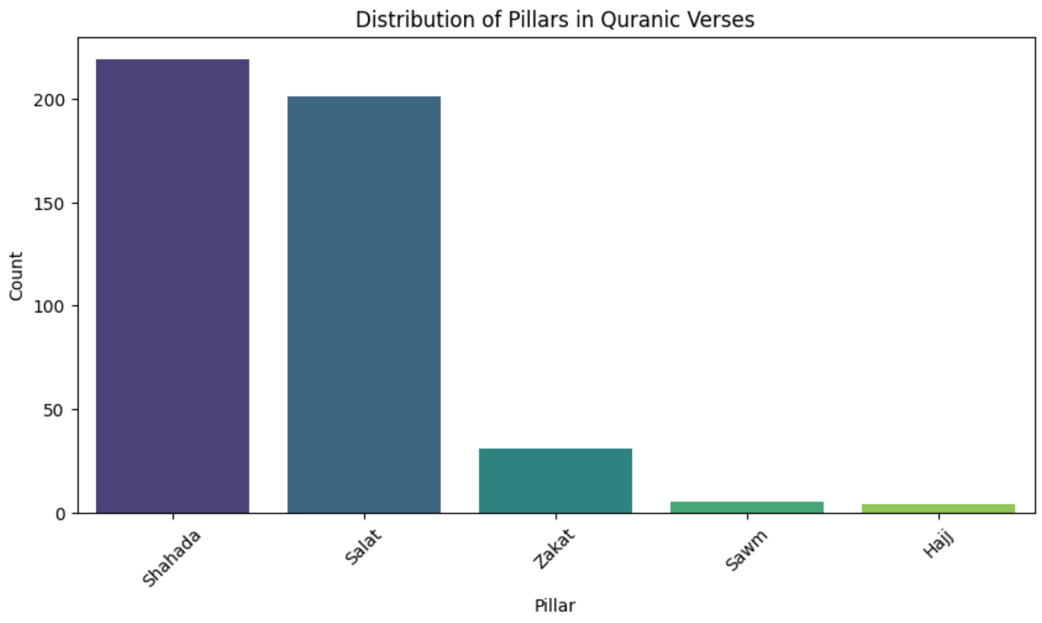
```
quran_data.head(4)
```

	Surah Number	Surah Name	Ayah Number	Ayah Text	Pillar
4	1	Text, Audio, Search, Download	5	It is You we worship, and upon You we call for...	Salat
10	2	al-Baqarah	4	Those who believe in the unseen, and perform t...	Salat
				O people! Worship your Lord who created	

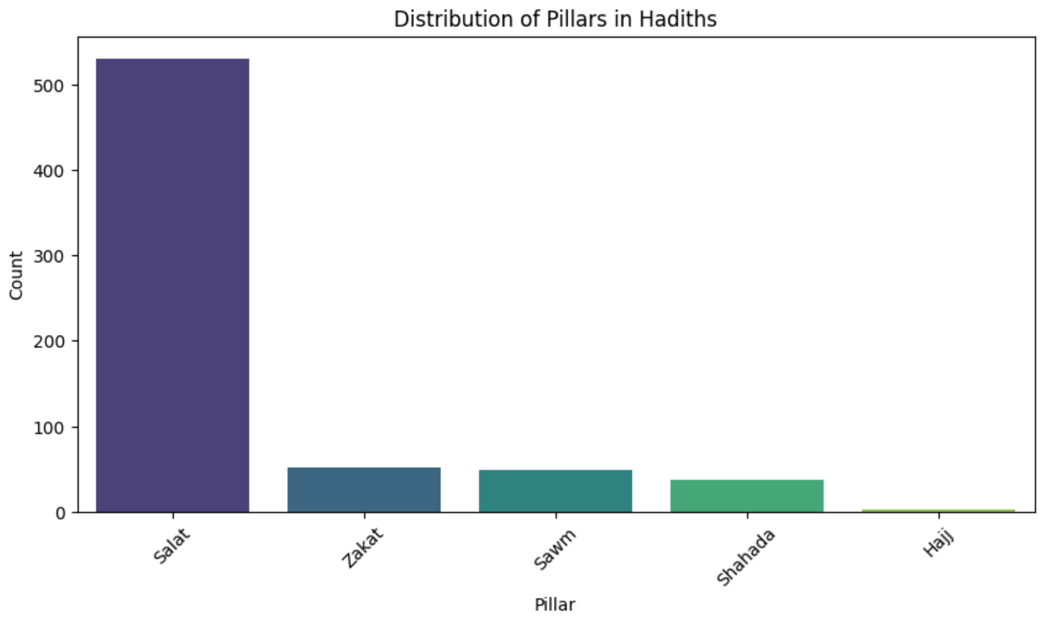
```
# Preprocess the documents using simple_preprocess from gensim
def preprocess(text):
    tokens = word_tokenize(text)
    tokens = [word.lower() for word in tokens if word.isalpha() and word.lower() not in stop_words]
    return tokens

# Preprocess the documents
quran_data['Preprocessed Text'] = quran_data['Ayah Text'].apply(preprocess)

# Create a dictionary from the preprocessed documents
dictionary = corpora.Dictionary(quran_data['Preprocessed Text'])

# Create a document-term matrix
doc_term_matrix = [dictionary.doc2bow(doc) for doc in quran_data['Preprocessed Text']]

# Apply LDA model
num_topics = 5
lda_model = gensim.models.LdaModel(doc_term_matrix, num_topics=num_topics, id2word=dictionary, passes=15)

# Pillar-topic mapping
pillar_topic_mapping = {
    0: "Salat",
    1: "Shahada",
    2: "Sawm",
    3: "Zakat",
    4: "Hajj"
}

# Print the topics and their keywords
for idx, topic in lda_model.print_topics(-1):
    pillar = pillar_topic_mapping.get(idx, "Unknown")
    keywords = ", ".join(word for word in topic.split('"') if word.isalpha())
    print(f"Pillar: {pillar}")
    print(f"Keywords: {keywords}\n")
```

```
Pillar: Salat
Keywords: god, worship, give, charity, say, except, people, good, lord, faith

Pillar: Shahada
Keywords: god, lord, witness, prayer, among, say, faith, said, worship, believe

Pillar: Sawm
Keywords: god, people, say, worship, said, may, us, faith, scripture, given

Pillar: Zakat
Keywords: god, believe, worship, whoever, disbelief, people, witness, day, lord, prayers

Pillar: Hajj
Keywords: god, witness, men, women, say, people, messenger, faithful, give, said
```

```
# Create bar graph for Quran data
pillar_counts = quran_data['Pillar'].value_counts()
plt.figure(figsize=(10, 5))
sns.barplot(x=pillar_counts.index, y=pillar_counts.values, palette='viridis')
plt.title('Distribution of Pillars in Quranic Verses')
plt.xlabel('Pillar')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

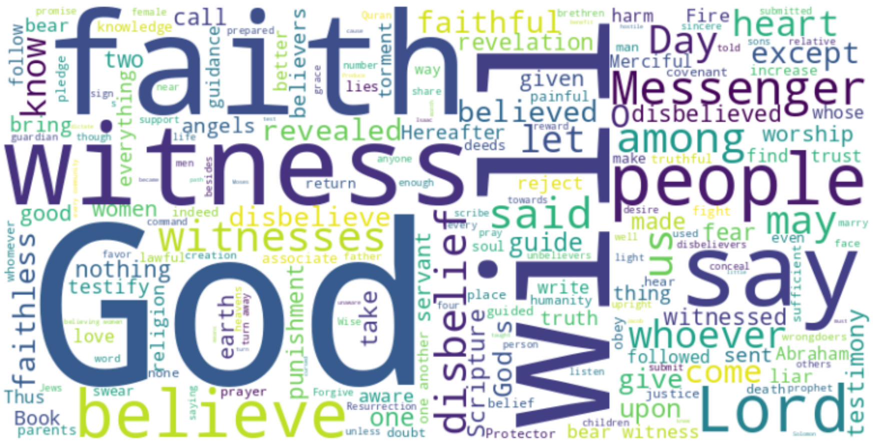


```
# Create bar graph for Hadith data
pillar_counts = hadith_data['Pillar'].value_counts()
plt.figure(figsize=(10, 5))
sns.barplot(x=pillar_counts.index, y=pillar_counts.values, palette='viridis')
plt.title('Distribution of Pillars in Hadiths')
plt.xlabel('Pillar')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

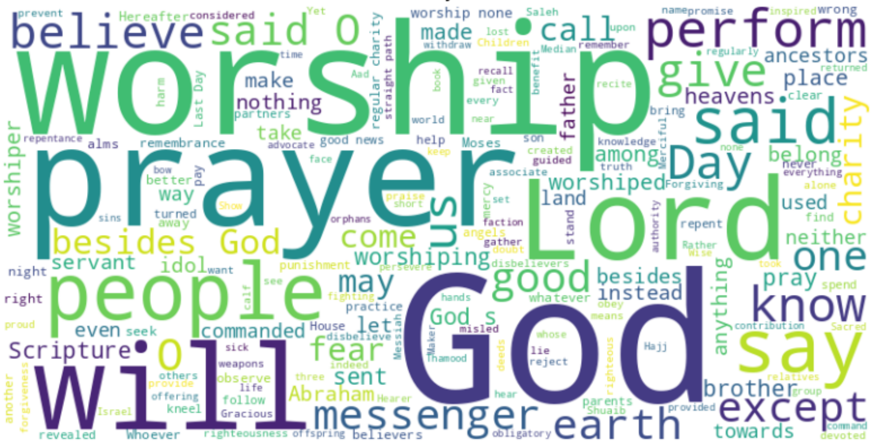


```
#word cloud for each pillar of quranic data
# Pillar-topic mapping
pillar_topic_mapping = {
    "Shahada": "Faith",
    "Salat": "Prayer",
    "Zakat": "Charity",
    "Sawm": "Fasting",
    "Hajj": "Pilgrimage"
}

# Create word cloud for Quran data
def create_quran_word_cloud(pillar_name):
    text = " ".join(quran_data[quran_data['Pillar'] == pillar_name]['Ayah Text'])
    wordcloud = WordCloud(width=800, height=400, background_color='white', colormap='viridis').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud for {pillar_topic_mapping[pillar_name]} Pillar (Quran)')
    plt.show()

# Create word clouds
for pillar_name in pillar_topic_mapping.keys():
    create_quran_word_cloud(pillar_name)
    print("\n\n\n")
```

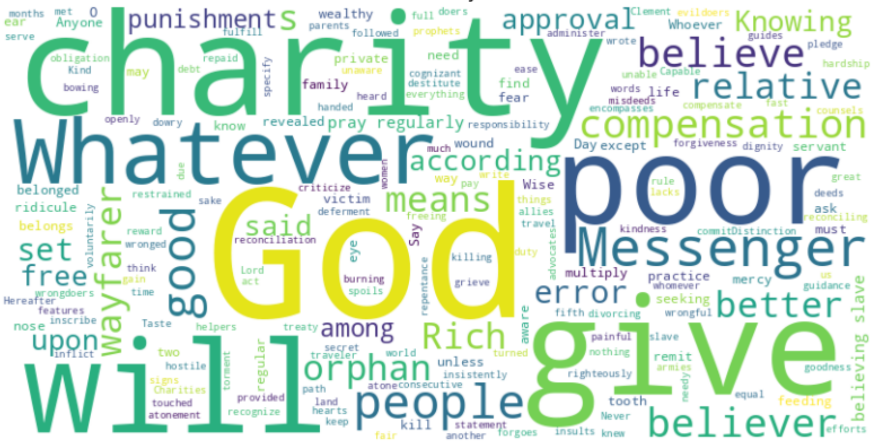
Word Cloud for Faith Pillar (Quran)



Word Cloud for Prayer Pillar (Quran)



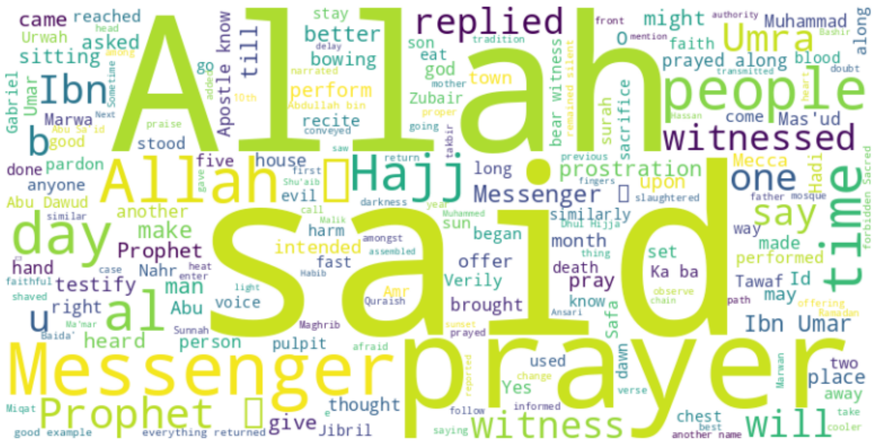
Word Cloud for Charity Pillar (Quran)



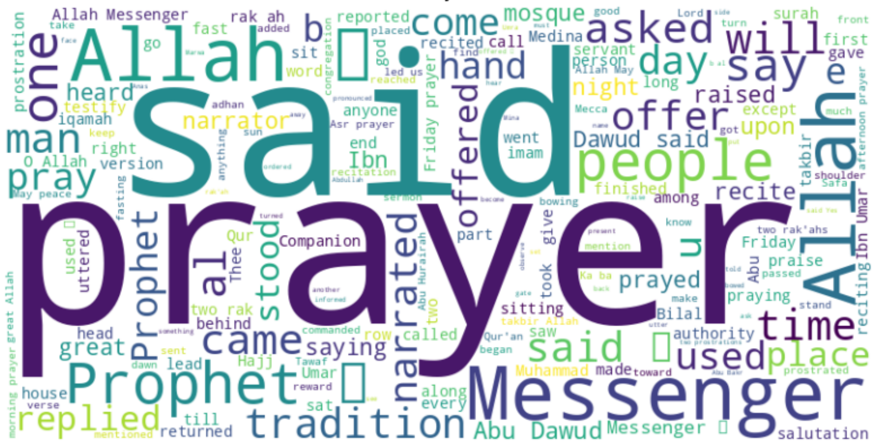
Word Cloud for Fasting Pillar (Quran)

```
#word cloud for each pillar of hadith data
# Pillar-topic mapping
pillar_topic_mapping = {
    "Shahada": "Faith",
     "Shahada": "Faith",
    "Salat": "Prayer",
    "Zakat": "Charity",
    "Sawm": "Fasting",
    "Hajj": "Pilgrimage"
}

# Create word cloud for Hadith data
def create_hadith_word_cloud(pillar_name):
    text = " ".join(hadith_data[hadith_data['Pillar'] == pillar_name]['Hadith Text'])
    wordcloud = WordCloud(width=800, height=400, background_color='white', colormap='viridis').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud for {pillar_topic_mapping[pillar_name]} Pillar (Hadith)')
    plt.show()

# Create word clouds
for pillar_name in pillar_topic_mapping.keys():
    create_hadith_word_cloud(pillar_name)
    print("\n\n\n")
```

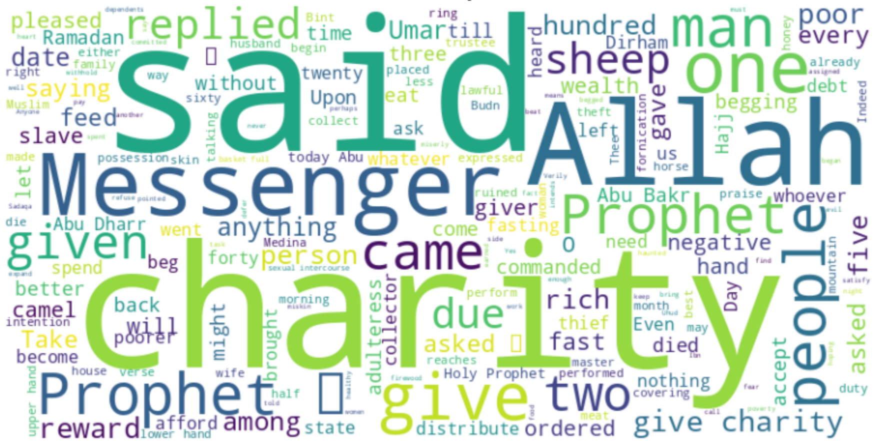
Word Cloud for Faith Pillar (Hadith)



Word Cloud for Prayer Pillar (Hadith)



Word Cloud for Charity Pillar (Hadith)



Word Cloud for Fasting Pillar (Hadith)

fast
day
Prophet
replied
one
till
say
hand
Al Wisal
Arafat
wife
ordered
performed
Hajj
fasted
year
give
revealed
overcast
saying
right

```
#search functionality
# Search function for Quran data
def search_quran(query):
    results = quran_data[quran_data['Ayah Text'].str.contains(query, case=False)]
    return results

# Search function for Hadith data
def search_hadith(query):
    results = hadith_data[hadith_data['Hadith Text'].str.contains(query, case=False)]
    return results

# Take user input for search query
user_query = input("Enter your search query: ")

# Perform search and print results
quran_results = search_quran(user_query)
print(f"Quran Search Results for '{user_query}':")
print(quran_results[['Ayah Text']])

print("\n")

hadith_results = search_hadith(user_query)
print(f"Hadith Search Results for '{user_query}':")
print(hadith_results[['Hadith Text']])
```

```
Enter your search query: faith
Quran Search Results for 'faith':
                                              Ayah Text
95    And they said, “Our hearts are sealed.” Rather...
100   And We made a covenant with you, and raised th...
105   Whoever is hostile to God, and His angels, and...
109   And they followed what the devils taught durin...
115   Or do you want to question your Messenger as M...
...                                                 ...
5597  We have appointed only angels to be wardens of...
5668  We have prepared for the faithless chains, and...
5787  We have warned you of a near punishment—the Da...
5877              These are the faithless, the vicious.
6230  They were commanded only to worship God, devot...

[93 rows x 1 columns]


Hadith Search Results for 'faith':
                                            Hadith Text
38    There are five thing, if anyone observe them w...
592   if anyone would like to have the fullest measu...
750   Marwan brought out the pulpit on 'Id. He began...
1009  (the mother of the faithful believers) I said,...
1020  When these two towns (Basra and Kufa) were cap...
1259  The Prophet (ﷺ) said, "Whoever established pra...
```

✓ 4s completed at 3:17 AM