Django Wordle Database Setup Guide

Overview

This guide explains how to set up the word database for your Django Wordle game using CSV word lists from Kaggle.

Prerequisites

- Django project already set up with models.py configured
- CSV files containing valid Wordle words
- Python environment with Django installed

Step 1: Obtain Word Lists from Kaggle

Option A: Using Separate Files

Download these files from Kaggle Wordle datasets:

- `valid_solutions.csv` - Contains 2,315 possible answer words
- `valid_guesses.csv` - Contains all valid guessable words (~12,972 words)

Option B: Using Combined File

Use the provided `combined_words.csv` which contains both valid solutions and guesses merged together.

Step 2: Prepare the Database Loading Script

Create a file called `load_words.py` in your Django project root directory:

```
import csv
import os
import django
from django.conf import settings

# Set up Django environment
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'wordgame.settings')
django.setup()

from game.models import Word

def load_words_from_csv(csv_path):
    """
    Load words from CSV file into the database.
    Args:
        csv_path (str): Path to the CSV file containing words
    """

    # Check if file exists
    if not os.path.exists(csv_path):
        print(f"Error: File '{csv_path}' not found!")
        print("Please make sure the CSV file is in the correct path.")
        return

    # Delete all existing words
    print("Deleting all existing words...")
    deleted_count = Word.objects.all().delete()[0]
    print(f"Deleted {deleted_count} existing words")

    # Load words from CSV
    print(f"Loading words from {csv_path}...")
    words_added = 0
    words_skipped = 0

    try:
        with open(csv_path, 'r', encoding='utf-8') as file:
            csv_reader = csv.reader(file)

            # Skip header if it exists
            first_row = next(csv_reader, None)
            if first_row and first_row[0].lower() in ['word', 'words', 'solution', 'solut
                print("Skipping header row")
            else:
                # If first row is not a header, process it
                if first_row and len(first_row[0]) == 5 and first_row[0].isalpha():
                    word = first_row[0].upper().strip()
                    Word.objects.create(word=word)
                    words_added += 1
                    if words_added <= 10:  # Show first 10 words
                        print(f"Added: {word}")

            # Process remaining rows
            for row in csv_reader:
                if row:  # Skip empty rows
                    word = row[0].upper().strip()

                    # Validate word (5 letters, alphabetic)
                    if len(word) == 5 and word.isalpha():
                        try:
                            Word.objects.create(word=word)
                            words_added += 1
                            if words_added <= 10:  # Show first 10 words
                                print(f"Added: {word}")
                            elif words_added == 11:
                                print("... (showing first 10 words only)")
                        except Exception as e:
                            print(f"Error adding word '{word}': {e}")
                            words_skipped += 1
                    else:
                        words_skipped += 1
                    if words_skipped <= 5:  # Show first 5 invalid words
                        print(f"Skipped invalid word: '{word}'")

    except FileNotFoundError:
        print(f"Error: Could not find file '{csv_path}'")
        return
    except Exception as e:
        print(f"Error reading CSV file: {e}")
        return

    # Final summary
    total_words = Word.objects.count()
    print(f"\n--- Summary ---")
    print(f"Words added: {words_added}")
    print(f"Words skipped: {words_skipped}")
    print(f"Total words in database: {total_words}")
    print("\nScript completed successfully!")

if __name__ == "__main__":
    # You can change this path to your CSV file
    CSV_PATH = 'combined_words.csv'  # or 'valid_solutions.csv'
    load_words_from_csv(CSV_PATH)
```

Step 3: Run the Database Setup

Method 1: Using the Script Directly

1. Place your CSV file in the project root directory
2. Run the script:

   ```
   python load_words.py
   ```

Method 2: Using Django Management Command

Create a custom management command for better integration:

1. Create directory structure:

   ```
   game/
   └── management/
       ├── __init__.py
       └── commands/
           ├── __init__.py
           └── load_words.py
   ```

2. Create `game/management/commands/load_words.py`:

   ```
   from django.core.management.base import BaseCommand
   from game.models import Word
   import csv
   import os
   ```

```
class Command(BaseCommand):
    help = 'Load words from CSV file into database'

    def add_arguments(self, parser):
        parser.add_argument('csv_file', type=str, help='Path to CSV file')
        parser.add_argument(
            '--clear',
            action='store_true',
            help='Clear existing words before loading',
        )

    def handle(self, *args, **options):
        csv_file = options['csv_file']

        if options['clear']:
            self.stdout.write('Clearing existing words...')
            deleted_count = Word.objects.all().delete()[0]
            self.stdout.write(f'Deleted {deleted_count} words')

        # Load words logic here (similar to above)
        # ... (implement the loading logic)
```

3. Run the command:

```
python manage.py load_words combined_words.csv --clear
```

Step 4: Verify the Database

Run this verification script to check if words were loaded correctly:

```
from game.models import Word

# Check total count
total_words = Word.objects.count()
print(f"Total words in database: {total_words}")

# Show some sample words
sample_words = Word.objects.all()[:10]
print("\nSample words:")
for word in sample_words:
    print(f"  {word.word}")

# Test the random word function
random_word = Word.get_random_word()
print(f"\nRandom word: {random_word}")
```

Step 5: Run Database Migrations

Ensure your database schema is up to date:

```
python manage.py makemigrations
python manage.py migrate
```

Troubleshooting

Common Issues:

1. **File not found**: Ensure CSV file is in the correct path
2. **Duplicate words**: The script handles duplicates by skipping them
3. **Invalid words**: Words that aren't exactly 5 letters or contain non-alphabetic characters are skipped
4. **Database locked**: Close any database connections before running the script

File Format Requirements:

- CSV file should have words in the first column
- Each word should be exactly 5 letters
- Words should contain only alphabetic characters
- Header row is optional (will be auto-detected)

CSV File Structure

Expected format for your CSV files:

```
word
ABACK
ABASE
ABATE
ABBEY
ABBOT
...
```

Or without header:

```
ABACK
ABASE
ABATE
ABBEY
ABBOT
...
```

Performance Notes

- Loading ~13,000 words typically takes 10-30 seconds
- The script shows progress for the first 10 words loaded
- Database operations are optimized for bulk insertion
- Consider using database transactions for even better performance

Next Steps

After successful database setup:

1. Test the game functionality
2. Verify random word selection works
3. Check that word validation works correctly
4. Run your Django server to test the complete game